Genome sequence-based virus taxonomy using machine learning

Tingting Wang

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

November 1, 2017

I, Tingting Wang, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Virus taxonomy is the task of partitioning the world of viruses into a coherent scheme of easily recognisable entities, with the major purpose of answering the everyday needs of practising virologists. Traditional approaches involve a lengthy process, done case by case through proposals by experienced virologists. With rapid advances in sequencing technology generating large numbers of virus genome sequences at an ever increasing rate, genome sequences are often the only information available for a virus in many situations. Traditional approaches are unable to handle this tsunami of data and to incorporate the newly identified viruses into existing systems in a timely and efficient manner.

Thus, automated methods for classifying viruses given only the primary structure of genomes are needed to aid the work of taxonomists. This thesis contributes to the application of machine learning techniques to genome sequence-based virus taxonomy. Specifically, we apply machine learning techniques to classify the NCBI reference sequences of virus model species into seven Baltimore Classes, four host groups or hundreds of ICTV hierarchical classes. We provide visualisations of a virus genome sequence dataset using various techniques and highlight properties of composition- and location-related nucleotide statistics, and statistics of the dataset as a whole. The thesis also provides a systematic experimental framework for applying machine learning techniques to virus taxonomy. Using the framework, we study the predictive power of various features of virus genome sequences and classifiers in multi-class classification, from simple single variable statistics to sophisticated high dimensional representations, from simple k-NN classifiers to more advanced SVM, RF and graph-based SSL methods. With optimised experimental factors, our

results outperform the current state of the art. In addition, we identify individual virus sequences that are frequently mislabelled by automated methods, study their memberships and provide predictions for currently unlabelled sequences using the best methods in our study. Finally, we extend the methods established in multi-class classification to the hierarchical classification problem of predicting ICTV taxonomic classes, which involves hundreds classes, many of them having very few samples per class. We find that both hierarchical and SSL approaches can improve performance in the task of virus genome classification.

Impact statement

This thesis contributes to the application of machine learning techniques to genome sequence-based virus taxonomy, with the ultimate aim of complementing traditional manual approaches.

The outputs of this thesis have significant impact both inside and outside academia, in the following ways. The methodologies used in this thesis contribute directly to automated virus taxonomy, and more broadly, research in bioinformatics and experimental machine learning. The features for virus genome sequences used in this thesis are applicable to other biological sequences and general discrete sequential data. The classifiers used are applicable to classification of both sequential and more general data types, and the experimental frameworks are useful for general applied machine learning studies.

Outside academia, the community of practical virologists will benefit most from the output of this thesis. They traditionally rely on laborious manual work to classify viruses. The automated taxonomy methods developed can significantly improve the efficiency and effectiveness of their work. This thesis also benefits clinicians who are dedicated to developing nucleotide sequence-based disease prevention and diagnostic procedures, by providing insights into the characteristics of genome sequences from different virus groups. With more effective procedures for epidemic prevention and disease diagnosis, the benefits can then be transferred to the general public. In addition, knowledge discovered in virus taxonomy contributes to a better understanding of biology and the life sciences, benefiting the education of the public.

Acknowledgements

First of all, I would like to highlight the significant contributions from my supervisors. Many thanks to my primary supervisor Dr. Mark Herbster for his support throughout my research, without whom my PhD would not have been possible. Mark is always professional as a scientist and patient as a mentor. He always encourages me to think from different perspectives by providing valuable insights into different aspects of machine learning. Thanks to my second supervisor Prof. Saira Mian, who was always present at our weekly supervisory meetings, providing useful biological insights and valuable advice, and was also willing to discuss ideas outside our meeting times.

I would also like to express gratitude to my viva examiners Prof. Nello Cristianini and Dr. Delmiro Fernandez-Reyes, and my transfer viva examiner Dr. Kevin Bryson for their professional discussions and valuable inputs into my thesis.

Thanks go to all my colleagues, friends and family, who accompanied and helped me during my PhD. Many thanks to the Computer Science Department at UCL and everyone who supported and trusted me at the beginning of my PhD when I entered the course straight from an undergraduate degree. Special thanks to Prof. Daniel Alexander, who gave me strong motivation to carry on and complete the PhD. Special thanks to Dr. Martin Parsley, who spent many days and nights going through my thesis and discussing revisions, and to Prof. Deyin Zhou, who generously shared his experience and thoughts. Thanks to Dr. Stephen Pasteris for his expertise in theoretical machine learning. A big thank you also goes to everyone in my family.

Contents

List of Figures

List of Tables

List of Abbreviations

CV	Cross-Validation
DT	Decision Tree
FC	Flat Classifier
GC	Global hierarchical Classifier
ICTV	International Committee on Taxonomy of Viruses
k-NN	k-Nearest Neighbour classifier
k-NNG	k-Nearest Neighbour Graph
LCL	Local hierarchical Classifiers per Level
LCN	Local hierarchical Classifiers per Node
LCPN	Local hierarchical Classifiers per Parent Node
MAW	Minimal Absent Word
MST	Minimum Spanning Tree
MV	Majority Vote
NCBI	National Center for Biotechnology Information
NCD	Normalised Compression Distance
NV	Natural Vector
RBF	Radial Basis Function
RF	Random Forest
RTD	Return Time Distribution
SL	Supervised Learning
SSL	Semi-Supervised Learning
SVM	Support Vector Machine
t-SNE	t-Distributed Stochastic Neighbour Embedding
USL	UnSupervised Learning

Chapter 1

Introduction

This chapter introduces the motivation for our work, the aims and contributions, and outlines the thesis.

1.1 Motivation

Virus taxonomy is the task of placing viruses into a taxonomic system, a scheme that groups organisms together on the basis of shared features such as evolutionary history, phenotypic characteristics, and biological properties [2].

Virus taxonomy has significant implications for both the scientific community and for practising clinicians. For the virologist, the classification of viruses provides a better insight into their biological properties. The task is challenging since the question of whether viruses are a type of life is already a controversial topic, and the taxonomical criteria established in the cellular kingdoms of life (Eubacteria, Archaea, Protista, Fungi, Plantae and Animalia) do not generally apply to the acellular kingdom of viruses [3, 2, 4]. The task is important to the understanding of life because it unveils new principles of life processes and promotes new directions in science [5, 6, 7].

For clinicians, virus taxonomy provides a reliable basis for medically significant differentiation, allowing for enhanced diagnostic procedures and epidemiological studies as well as facilitating the treatment and prevention of disease. For example, the introduction of diagnostic procedures based on nucleic acid sequences has increased the need for precise classification of viruses in accordance with their

molecular characteristics [8]. The identification of virus hosts has practical importance in the prevention of potentially large scale epidemics [9].

1.1.1 Virus taxonomic schemes

Virus taxonomic schemes evolve over time as virology progresses and technology develops. In the early days of virology when genome sequences were unavailable, viruses were classified and named after the host or location where they were first discovered. Such taxonomy is completely based on phenetic properties. Nowadays, host range is still an important property of a virus and is one of the criteria used for classification [2].

Modern taxonomy relies more heavily on the genetic properties of viruses. Currently, two main taxonomic systems are in use: the Baltimore scheme and the International Committee on Taxonomy of Viruses (ICTV) scheme. The Baltimore scheme [10] defines seven classes based on genome organisation and replication strategy, assigning a virus to Baltimore Class I, II, III, IV, VI, VI or VII according to its nucleic acid type (DNA or RNA), strandedness (single or double), genome polarity (positive-sense or negative-sense), and method of viral mRNA synthesis (reverse transcription or not). The ICTV scheme [2] defines a hierarchy based on multiple criteria, including the genome sequence and relationships to known viruses. A virus is assigned to a newly suggested or extant Order (most general), Family, Subfamily, Genus and Species (most specific) following proposals made by virologists. Viruses designated as the same Species have several properties in common but need not have a single common defining property. Those assigned to the same class at a higher (more general) level in the hierarchy share certain common properties from lower level classes.

1.1.2 The need for automated genome sequence-based techniques

Manually classifying a virus can be a laborious task and involves a lengthy process. It is typically done case by case through proposals by experienced virologists, who make significant efforts to understand the various biological properties

of certain virus groups [2]. With rapid advances in sequencing technology generating large numbers of virus genome sequences at an ever increasing rate, these genome sequences are often the only information available for a virus in many real world situations [4]. Manual approaches to virus taxonomy are unable to handle this tsunami of data and to incorporate the newly identified viruses into existing systems in a timely and efficient manner. Thus, automated methods for classifying viruses given only the primary structure of virus genomes are needed to aid the work of taxonomists.

Given the rich evolutionary history of the virus kingdom, a taxonomy based only on genome sequences may not properly reflect their phylogenetic properties and thereby yield fewer biological insights [4, 3]. However, the overall effectiveness and efficiency of such methods make them useful tools for taxonomy-related tasks, and it is broadly agreed that the development of a robust framework for sequence-based virus taxonomy is indispensable to the comprehensive characterisation of viruses [11, 12, 13]. Moreover, at a recent ICTV workshop, experts reached a consensus that with appropriate quality control, viruses that are known only from metagenomic data should be incorporated into the official ICTV classification scheme [14].

1.1.3 Methods for sequence comparison

In order to perform genome sequence-based virus taxonomy, methods for sequence comparison are required to establish the relationship between different viruses. In general, genome sequence-based taxonomy can be divided into two types: alignment-based and alignment-free methods.

Alignment-based methods Alignment-based methods are the traditional approach to sequence comparison, and classify viruses based on the degree to which their genome sequences can be matched. These methods first identify conservative regions in the genome sequences, align them through insertion, deletion and mutation, and then derive distance measures between the genomes using alignment scores.

Numerous such techniques exist, mainly differing in the way the sequences are aligned and/or the scores derived (see [15, 16] for detailed reviews). For example,

alignment can be performed between every two sequences [17, 18, 19] or between multiple ones simultaneously [20, 21, 22]; alignment can be performed based only on certain local structures of sequences [23, 24, 18] or on the global structure as a whole [25, 26, 27]. A wide range of scoring systems have also been proposed, such as the substitution scoring matrices PAM [28] and BLOSUM [29].

These methods perform well for relatively small datasets consisting of similar sequences, however can suffer from both computational and fundamental limitations on larger datasets with diverse sequences. In terms of computational load, optimal sequence alignment can be infeasible for the large corpora of sequences produced by next generation sequencing technologies [30, 31]. Alignment-based methods (e.g. [23, 21]) typically require time and space complexity on the order of $O(L^2)$, where L is the average length of the sequences. More efficient methods (e.g. [32, 33]) have been developed for specialised purposes, but typically assume specific properties for the sequences being aligned. In terms of virological fundamentals, the evolutionary assumptions guiding the alignment and scoring procedures may not reflect the phylogeny, tending to overemphasize the importance of sequence similarity while overlooking the importance of functional similarity [34, 35]. In addition, the scoring methods assume linearity of the evolutionary procedure, which, in fact, takes place at different scales simultaneously [36]. Moreover, due to the lack of a feature representation, they can only be combined with distance-based classifiers, which restrict the application of potentially more sophisticated and powerful machine learning techniques.

Alignment-free methods In contrast, alignment-free methods – the focus of this thesis, classify viruses based on the degree to which the features of different sequences are similar. Instead of aligning sequences or deriving similarity scores, they first map a virus genome sequence to a point in feature space, where the distance between features reflects the distance between the original sequences, then classify the virus in this space.

Alignment-free methods were initiated by [37]. Recent representations and approaches include using the statistics of nucleotide occurrence and positional in-

formation [38, 39], count of k-mers [40], Kolmogorov complexity of sequences [41], absent words [42, 43], matrix invariants [44], genomic signal processing [45], curves [46] and images [47].

In relying solely on the analysis of the genome sequence, alignment-free methods suffer from the same drawbacks as alignment-based methods, lacking biological insights into the functionalities of their viruses. However, the former improve upon the latter in several aspects. Firstly, since no alignment is needed, the associated biological knowledge required to inform the alignment process is not required, which can be an advantage in situations where only the sequence is known. Secondly, they can cope with highly diverse sequences where reliable alignment is impossible [11]. Thirdly, they can handle large data sets more efficiently as all sequences are represented in a fixed format as points in a feature space. Furthermore, the usage of features allows for the application of a wider range of machine learning techniques, such as the k-NN classifier [38, 48], association rule-based classifier [49], SVM [50] and artificial neural networks [51].

An earlier work [52] shows that alignment-free methods using nucleotide statistics perform well for diverse virus genome sequences (above Genus level) but tend to become less accurate for similar ones (Genus and Species levels). However, later studies [48, 40] show that with more sophisticated features, alignment-free methods perform no worse than alignment-based ones even at Genus and Species levels. A recent work has proposed a strategy that exploits the complementary nature of alignment-based and alignment-free methods [53]. The classification of sequences is performed by using a combined sequence similarity score (CSSS) that is calculated based on the weighted contribution of the scores of each method, where the weights reflect the discriminatory ability of individual measures in the training set. Scores combined from 3 alignment-free and 2 alignment-based methods show good performance in biological sequences.

1.2 Aims and contributions

The goal of this thesis is to advance automated virus taxonomy methods using only the primary structure of virus genomes, with the ultimate aim of complementing traditional manual approaches. We focus on alignment-free methods for sequence comparison, studying the performance of different features for genome sequences and machine learning techniques in the task of virus taxonomy.

Our aims are as follows. First, we will asses the extent to which information provided by genome sequences can be used to distinguish viruses from different taxonomic classes. Next, we will investigate the predictive powers of different feature representations of genome sequences and classifiers in the task of virus taxonomy. Finally, we will devise classification strategies that outperform the current state of the art.

Our contributions are as follows. First, we conduct a thorough analysis of the NCBI reference genome sequence dataset that is frequently used by virologists, exceeding previous studies in scale and depth (Chapter 5). We analyse statistical properties of the nucleotide composition of genome sequences for all currently discovered virus species, which provides insights into the efficacy of genome sequence-based taxonomy. We also analyse the predictive powers of different feature representations for genomes based on their statistical properties. In addition, we apply visualisation techniques to display all the sequences in the dataset in two-dimensional space. Second, we conduct a systematic study to compare and contrast the predictive powers of various combinations of features and classifiers in the task of virus taxonomy, extending previous studies with a wider range of techniques from simple approaches to sophisticated ones (Chapter 6 and 7). With optimised experimental factors, the best combinations outperform the current state of the art. Using the best methods identified in our study, we make predictions for currently unlabelled sequences. Third, we are the first to explicitly incorporate hierarchical information and apply SSL-based hierarchical classifiers to the dataset in predicting the ICTV classes (Chapter 8). The SSL-based hierarchical methods outperform SL-based ones, which outperform non-hierarchical ones.

1.3 Thesis outline

The remainder of the thesis is organised as follows.

In Chapter 2, we describe the virus genome sequence dataset and taxonomic scheme used in our study, as well as presenting the summary statistics of the dataset.

In Chapters 3 and 4, we review machine learning techniques for sequence classification and feature representations for genome sequences. In Chapter 3, we review the machine learning techniques used in our study. We start by categorising learning techniques, comparing and contrasting different approaches. We then review the classifiers used in our study, including k-NN, RF, SVM, and Graph-based SSL. Following this, we review hierarchical classifiers, including their characterisations, approaches, relationships with each other as well as their bioinformatics applications. Finally, we discuss performance measures used for non-hierarchical and hierarchical classification.

In Chapter 4, we review the alignment-free feature representations used in our study. We first describe nucleotide-based features, including NV and its derivatives. We then describe word-based features, which include k-mer count, k-mer NV, RTD and MAW. In addition, we describe compression-based features derived from both general and DNA-specific compression tools.

Chapters 5 to 8 contain our key observations, results and contributions. In Chapter 5, we perform a thorough exploratory analysis of the the entire virus genome sequence dataset, exceeding prior work in scale and depth. The analysis aims to better understand the properties of the dataset, as well as informing hypothesis formulation and experimental design in later chapters. We explore different properties of composition- and location-related nucleotide statistics, word and compression-based features, as well as the dataset as a whole using various visualisation techniques, including ternary plots and t-SNE visualisations.

In Chapter 6, we extend previous work by performing a systematic study of the classification performance of the NV and its derivatives. We study the predictive power of NV and its components combined with various machine learning techniques, from the simple 1-NN and k-NN classifiers used in previous works, to

more advanced SVM and RF classifiers that have not been explored before. Using the best experimental settings, we identify viruses that are consistently misclassified and predict labels for currently unlabelled sequences. We also investigate the performance in predicting virus hosts.

In Chapter 7, we extend the nucleotide-based features used in Chapter 6 to more sophisticated word and compression-based features. We perform the first systematic study of the classification performance of a wide range of features and classifiers. As in Chapter 6, using the best experimental settings, we identify viruses that are consistently misclassified and predict labels for currently unlabelled sequences. We also investigate the performance in predicting virus hosts and taxonomic classes for multi-segmented viruses.

In Chapter 8, we design hierarchical classification approaches based on the best non-hierarchical classifier identified in Chapter 7 to classify viruses into hierarchical ICTV taxonomic classes. We are the first to explicitly incorporate ICTV hierarchical information into classification, and also the first to apply SSL-based hierarchical classifiers to the classification of ICTV taxonomic classes.

In Chapter 9, we conclude our study and discuss directions for future work.

Chapter 2

Virus genome sequence datasets

This chapter describes the types of public virus genome sequence datasets and provides an overview of the dataset used in our study. We use a genome sequence dataset provided by the National Center for Biotechnology Information (NCBI), which is part of the United States National Library of Medicine, a branch of the National Institutes of Health. The NCBI houses a series of databases relevant to biotechnology and biomedicine and is an important resource for bioinformatics tools and services.

2.1 Types of datasets

The NCBI provides two types of virus genome sequence datasets: a reference sequence dataset that contains a single record for each model virus species, and a comprehensive dataset that contains all publicly available records for individual viruses. They are available through the Reference Sequence database (RefSeq) [54] and the GenBank database [55] respectively.

2.1.1 RefSeq

The RefSeq database is an open access, annotated and curated collection of publicly available nucleotide sequences and their protein products. The collection aims to provide, for each model species, a complete set of non-redundant, extensively cross-linked, and richly annotated nucleic acid and protein records. Each record represents a synthesis of the primary information of a species that was generated and submitted by different research groups. The database is curated on an ongoing

basis by collaborating groups and NCBI staff. Sequence records are presented in a standard format and subjected to computational validation. The collection establishes a useful baseline for integrating diverse data types.

2.1.2 GenBank

The GenBank database is also an open access collection of nucleotide sequences and protein products. In contrast to the RefSeq database, its aim is to provide a comprehensive collection of all original sequences. Each record represents the primary information of an individual virus submitted by a group, by whom the copyright is retained. The originality and quality of the sequences are checked by the NCBI staff, but the sequences themselves are not curated. The collection establishes a comprehensive archive for discovered and approved sequences.

2.2 Summary of the experimental dataset

Since the theme of the thesis is to investigate the effectiveness of genome sequence-based techniques in the task of virus taxonomy, we are interested in genome sequences representative of virus species rather than individuals. Therefore, we use the RefSeq dataset as it contains high quality and complete model genomes for each virus species.

The RefSeq dataset was retrieved from the "Viruses" directory of the NCBI FTP site on 18^{th} September 2015 [56]. Each of the 4,420 downloaded folders contains the complete genome sequence and annotation for a virus species. A virus genome can consist of a single or multiple nucleotide segments. The dataset contains 3,910 non-segmented viruses (of which 211 are satellites) and 510 multi-segmented viruses (of which 6 are satellites). We remove all satellites from the dataset, leaving genome sequence data for the 3,699 non-satellite non-segmented viruses and 504 non-satellite multi-segmented viruses for later experiments. We focus our study on non-segmented viruses since the number of multi-segmented viruses is small. Virus host labels are not available from the "Viruses" directory, which we downloaded separately from http://www.ncbi.nlm.nih.gov/genome/browse/.

2.3 Taxonomic schemes

We consider three different taxonomic schemes with decreasing levels of dependence on genome sequences: ICTV Orders, Baltimore Classes and virus hosts. The Baltimore scheme classifies viruses into 7 non-hierarchical classes whereas the ICTV scheme defines a hierarchy starting at 7 Orders and progressing through 77 Families, 19 Subfamilies and finally 371 Genera. The viruses are also assigned 14 different host labels, which we reduce to 4 non-hierarchical classes Archaea, Bacteria, Eucaryote 1 and Eucaryote 2.

2.3.1 Non-hierarchical classes

Table 2.1 breaks down the 3,699 non-segmented and the 504 multi-segmented viruses investigated by Baltimore Class and ICTV Order. Table 2.2 and 2.3 show their membership in the two schemes. They reveal that an ICTV Order is a subgroup of a Baltimore Class, i.e. each Baltimore Class contains multiple ICTV Orders but each ICTV Order belongs to only one Baltimore Class. Table 2.4 breaks down the 3,699 non-segmented viruses by their hosts. The fourteen host labels are divided into four groups in order to balance the class sizes and avoid overlaps in constituting hosts.

In our non-hierarchical classification experiments of non-segmented viruses, we classify a virus genome sequence into one of the 7 Baltimore Classes, one of the 7 ICTV Orders or one of the 4 host groups. Due to the small number of labels available for multi-segmented viruses, our experiments classify each of them into one of the 4 Baltimore Classes II, III, IV or V. Table 2.5 summarizes the entropy of the classification problem for each taxonomic scheme.

Baltimore Class	**Abbreviation**	**Number of sequences**	
		Non-segmented	Multi-segmented
I: dsDNA	I	1,861	5
II: ssDNA	II	556	132
III: dsRNA	III	83	123
IV: (+)ssRNA	IV	834	138
V: (-)ssRNA	V	157	93
VI: ssRNA-RT	VI	38	0
VII: dsDNA-RT	VII	96	0
Labelled		3,625	491
Unlabelled		74	13
Total		3,699	504

ICTV Order	**Abbreviation**	**Number of sequences**	
		Non-segmented	Multi-segmented
Caudovirales	C	1,281	0
Herpesvirales	H	66	0
Ligamenvirales	L	13	0
Mononegavirales	M	156	0
Nidovirales	N	67	0
Picornavirales	P	135	38
Tymovirales	T	147	0
Labelled		1,865	38
Unlabelled		1,834	466
Total		3,699	504

Table 2.1: The non-satellite non-segmented virus genome sequences labelled by Baltimore Class and ICTV Order.

For each taxonomic scheme, the complete name of a Baltimore Class or ICTV Order, its abbreviation, and the number of sequences assigned that taxonomic label are listed. The row "Labelled" shows the number of sequences assigned any one of the seven labels. "Unlabelled" is the number of sequences with no label. "Total" is the number of labelled and unlabelled sequences. For the Baltimore scheme, "ds" denotes double-stranded, "ss" single-stranded, "(+)" positive-sense, "(-)" negative-sense, and "RT" reverse-transcribed.

	I	II	III	IV	V	VI	VII	Unlabelled
C	1281	0	0	0	0	0	0	0
H	66	0	0	0	0	0	0	0
L	13	0	0	0	0	0	0	0
M	0	0	0	0	156	0	0	0
N	0	0	0	67	0	0	0	0
P	0	0	0	135	0	0	0	0
T	0	0	0	147	0	0	0	0
Unlabelled	501	556	83	485	1	38	96	74

Table 2.2: Membership of non-satellite non-segmented viruses in the Baltimore Class and ICTV Order schemes.

For each Baltimore Class (column), the number of viruses assigned that taxonomic label and an ICTV Order (row) or no additional label (row "Unlabelled") is shown. For example, the entry corresponding to the row "H" and column "I" means there are 66 viruses from Baltimore Class "I" and ICTV Order "H". The table shows abbreviations of class names, see Table 2.1 for the full names.

	I	II	III	IV	V	Unlabelled
P	0	0	0	38	0	0
Unlabelled	5	132	123	100	93	13

Table 2.3: Membership of non-satellite multi-segmented viruses in the Baltimore Class and ICTV Order schemes.

The same as Table 2.2 but for non-satellite multi-segmented viruses. Missing Baltimore Classes and ICTV Orders correspond to columns and rows consisting of zeros only.

Host group	**Abbreviation**	**Hosts**	**Number of sequences**	**Total**
Archaea	A	Archaea	63	63
Bacteria	B	Bacteria	1,421	1,421
Eucaryote 1	E1	Algae Fungi Protozoa	40 67 30	137
Eucaryote 2	E2	Invertebrates Invertebrates, plants Invertebrates, vertebrates Plants Vertebrates Vertebrates, human Vertebrates, invertebrates Vertebrates, invertebrates, human	234 38 4 726 736 215 23 47	2,023
Subtotal				3,644
Environment				43
Labelled				3,687
Unlabelled				12
Total				3,699

Table 2.4: The non-satellite non-segmented virus genome sequences labelled by hosts.

The fourteen virus host labels obtained from the dataset, excluding Environment, are divided into four groups: Archaea (A), Bacteria (B), Eucaryote 1 (E1) and Eucaryote 2 (E2). For each host group, its name, abbreviation, constituent hosts, the number of sequences assigned to that host, and the total number in the group are shown. Row "Subtotal" is the number of sequences used in our experiments. "Environment" is the number of sequences assigned Environment as host, which are excluded from our experiments. "Labelled" is the number of sequences assigned any one of the four labels. "Unlabelled" is the number of sequences with no host labels. "Total" is the number of all the sequences listed in the table.

Segment type	Taxonomic scheme	Possible classes	Entropy
Non-segmented	Baltimore Class	I, II, III, IV, V, VI, VII	0.252
Non-segmented	ICTV Order	C, H, L, M, N, P, T	0.432
Non-segmented	Host group	A, B, E1, E2	0.111
Multi-segmented	Baltimore Class	II, III, IV, V	0.117

Table 2.5: Entropy of virus classification for each taxonomic scheme.

The table shows virus segment type, taxonomic scheme used, possible classes in the classification problem and entropy. Entropy is computed as $H(X) = -\sum_{i=1}^{c} p_i ln(p_i)$, where c is the number of possible classes in the classification problem, ln is the natural log, and p_i is the probability of a virus coming from class i, which we assign to the inverse of the number of total labelled viruses in that class.

2.3.2 Hierarchical classes

The ICTV scheme organizes the classes as a hierarchical taxonomic tree, and from the highest to the deepest levels are Order, Family, Subfamily, Genus and Species respectively. The tree is balanced where every leaf node is at exactly the same depth. Table 2.6 shows the labelling rate at each ICTV hierarchical level for the 3,699 non-satellite non-segmented viruses in the dataset. Table 2.7 summarises the labelling conditions for individual sequences of model species. Among them, 221 species are not assigned to any high level classes and only 288 have labels at all levels. Our hierarchical classification experiments aim to assign the genome sequence of each model Species in the dataset into the Order, Family, Subfamily and Genus that it belongs to.

Taxonomic level	Number of classes	Number of labelled genomes	Labelling rate
Order	7	1865	0.504
Family	77	3403	0.920
Subfamily	19	553	0.149
Genus	371	2368	0.640
Species	3699	3699	1

Table 2.6: Labelling rate at each ICTV hierarchical level.

The table shows the number of classes, number of labelled genomes and labelling rate (the proportion of viruses having a label at the given level) at each ICTV taxonomic level.

nLabels	Order	Family	Subfamily	Genus	nSpecies
0					221
1	x				30
1		x			158
1			x		0
1				x	41
2	x	x			832
2	x		x		0
2	x			x	4
2		x	x		11
2		x		x	1228
2			x	x	0
3	x	x	x		79
3	x	x		x	632
3	x		x	x	0
3		x	x	x	175
4	x	x	x	x	288
Total	3699	1865	3403	553	2368

Table 2.7: ICTV label assignment for non-satellite non-segmented viruses in the dataset.

The column “nLabels” shows the number of higher level labels a genome sequence of a model species has. The columns “Order”, “Family”, “Subfamily” and “Genus” show the levels of the present labels. The corresponding level is marked with an “x” if a label is present. The last column “nSpecies” shows the number of species with the given “nLabels”. The last row “Total” shows the total number of viruses in the dataset, as well as the total number of those labelled at corresponding levels.

Chapter 3

Machine learning techniques

Machine learning techniques are computational algorithms that are able to extract rules from training data (learning) and apply these to testing data (prediction) without being explicitly programmed [57, 58, 59]. Among other uses, they can automate manual tasks to aid human experts in the analysis of large and complex data sets. In genomics, machine learning is perhaps most useful for the interpretation of large genomic data sets and has been used to annotate a wide variety of genomic sequence elements [60].

This chapter reviews the categories of machine learning techniques, the non-hierarchical and hierarchical classifiers and performance measures we use in our study.

3.1 Categories of machine learning techniques

Machine learning techniques can be categorized in a number of different ways. For instance, depending on whether each training sample is assigned a class label, they can be divided into supervised, unsupervised and semi-supervised methods. They can also be divided depending on whether distance matrices or feature vectors are required as the input, or they can be divided into distance-based and feature-based methods, or depending on whether the classes have a hierarchical relationship between each other, divided into non-hierarchical and hierarchical methods.

3.1.1 Supervised vs unsupervised vs semi-supervised

Machine learning techniques can be divided into three main classes based on the presence of labels when building models: supervised learning (SL), where labelling information of known data guides the learning process; unsupervised learning (USL), where rules are learned by the intrinsic properties of the data without the aid of labelling information; and semi-supervised learning (SSL), which combines labelled and unlabelled data during learning and prediction. When applied to virus taxonomy, SL techniques (such as classification) can be used to predict the taxa labels of a new virus. On the other hand, USL techniques (such as clustering) can be used to explore natural groups of viruses and construct phylogenetic trees. The third type, SSL techniques, can be used to improve classification or clustering in the situation where taxa labels are available for a small number of samples but are missing for the others. This thesis focuses on SL and SSL for classification.

3.1.2 Distance-based vs feature-based

Machine learning methods can be divided into three types based on the mechanisms they use to operate on distance measures and features. The first are distance-based methods, such as k-nearest neighbours (k-NN) and k-means. They do not require feature representations of the samples as long as their pairwise distance can be obtained. These methods take the distance matrix computed from all the samples as input and group nearby ones together. The second type are feature-based methods, such as Decision Trees (DT) and Random Forests (RF). In contrast to distance-based methods, feature representations of the original data are compulsory. These methods take the feature of a sample as input and assign labels by examining the discriminative power of the individual variables constituting each feature. The third type combine the above two, including Support Vector Machines (SVM) and other kernel methods. They require both a specific distance measure definition and feature representation. Kernel methods take features as input and implicitly transform them into a high-dimensional space through a kernel function, which plays a similar role to a distance measure.

The benefits of such a division can be clear when we discuss alignment-based

and alignment-free methods for sequence analysis: the lack of features restricts alignment-based methods to distance-based methods only, whereas the availability of features allows a wide range of classifiers to be combined with alignment-free methods. A brief discussion about distance and feature-based methods for sequence classification can be found in [61].

3.1.3 Non-hierarchical vs hierarchical

Machine learning methods can also be divided into non-hierarchical and hierarchical approaches depending on whether the different classes of samples have a hierarchical relationship with each other. Most machine learning techniques are not originally designed to address hierarchical relationships explicitly and they typically assume no hierarchical relationship between different classes. Hierarchical approaches are usually developed by incorporating hierarchical information into their non-hierarchical counterparts, which we call "base classifiers" in the context of hierarchical classification.

3.2 Non-hierarchical classifiers

We use three types of classifiers in our non-hierarchical classification experiments, k-NN, RF and SVM, representing distance-based methods, feature-based methods and those combining the two.

3.2.1 k-Nearest Neighbour (k-NN)

k-NN [62] is a classifier previously used in similar studies [52, 48]. To predict the label of a new virus genome sequence, it first computes the distance between the feature vector of the given sequence and that of all others in the training set, then makes a prediction using the majority vote of labels of the k-nearest neighbours (sequences having the closest distance to the given one), where k is a parameter of the model. This is implemented by the function `knn` [63] (Table 3.1).

3.2.2 Random Forest (RF)

RF [64] is an ensemble method consisting of a collection of Decision Trees. During training, a multitude of uncorrelated DT are constructed, with each tree built using

a random subset of virus genome sequences in the training set. To grow a tree, a random subset of feature variables are selected as candidates for node splitting, and the one that maximises a certain measure of information gain is used. The tree is then grown by repeatedly splitting nodes until termination conditions are met. To predict the label of a new sequence, each tree casts a unit vote for the predicted class and the one with the majority votes is the final output of the RF.

Function `randomForest` [65] (Table 3.1) is an interface to the program by Breiman and Cutler described in [64]. This builds a tree using two-thirds of the samples in the training set drawn randomly with replacement, splits each node of a tree using Gini impurity as the measure, then grows the tree using a CART (Classification and Regression Tree) methodology [66] until all samples in a node are from the same class. The trees grown are not pruned. The function has two tunable parameters. One is $ntree$, the number of trees in the forest and another is $mtry$, the number of variables randomly sampled for splitting a node. According to [64], the influence of the parameters are found to be small and a wide range of values tend to give optimal results.

3.2.3 Support Vector Machine (SVM)

The SVM [67, 68, 69] is a binary classifier that aims to find a separator that best separates samples from different classes. The optimization goal is to find a hyperplane that maximises the margin – the distance of the samples closest to the hyperplane. Given training samples $(\mathbf{x}_i, y_i), i = 1, ..., m$, where $\mathbf{x}_i \in R^d$ is a feature vector of dimension d and $y_i \in \{-1, 1\}$ is the label of the i-th sample, the primal form of the classical soft-margin problem is formulated as:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{m} \varepsilon_i, \tag{3.1}$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \varepsilon_i, \tag{3.2}$$

$$\varepsilon_i \geq 0, i = 1, ..., m. \tag{3.3}$$

A new sample $\mathbf{x}$ is classified as $sign(\hat{\mathbf{w}}^T\mathbf{x}+\hat{b})$, where $\hat{w}$ is the optimised parameter vector and $\hat{b} = y_j - \hat{\mathbf{w}}^T\mathbf{x}_j$. The dual form of the optimisation problem is

$$\max_{\alpha_i} \quad -\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j + \sum_i \alpha_i, \tag{3.4}$$

$$s.t. \quad \sum_i y_i\alpha_i = 0, \tag{3.5}$$

$$0 \leq \alpha_i \leq C, i = 1,...,m. \tag{3.6}$$

A new sample $\mathbf{x}$ is classified as $sign(\sum_{i=1}^{m}\hat{\alpha}_i y_i \mathbf{x}_i^T\mathbf{x}+\hat{b})$, where $\hat{\alpha}_i$ is the optimised parameter and $\hat{b} = y_j - \sum_{i=1}^{m}\hat{\alpha}_i y_i \mathbf{x}_i^T\mathbf{x}$.

The benefit of the dual form is that a feature map $\phi(.)$ can be introduced to enhance the separability of the data points. This is done by replacing the original feature vector $\mathbf{x}_i$ with $\phi(\mathbf{x}_i)$ and the corresponding similarity measure can be computed using a kernel function $K(\mathbf{x}_i,\mathbf{x}_j) =< \phi(\mathbf{x}_i),\phi(\mathbf{x}_j) >$. Then the above formulation becomes

$$\max_{\alpha_i} \quad -\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) + \sum_i \alpha_i, \tag{3.7}$$

$$s.t. \quad \sum_i y_i\alpha_i = 0, \tag{3.8}$$

$$0 \leq \alpha_i \leq C, i = 1,...,m. \tag{3.9}$$

A new sample $\mathbf{x}$ is classified as $sign(\sum_{i=1}^{m}\hat{\alpha}_i y_i K(\mathbf{x}_i,\mathbf{x})+\hat{b})$, where $\hat{\alpha}_i$ are the optimised parameters and $\hat{b} = y_j - \sum_{i=1}^{m}\hat{\alpha}_i y_i K(\mathbf{x}_i,\mathbf{x}_j)$.

The role of a kernel function is to map the training samples to a high-dimensional space, where the data can be linearly separated by the hyperplane. The choice of kernel function determines the type of space the data points are mapped to. The inner product between the original feature vector $\mathbf{x}_i^T\mathbf{x}_j$ can be considered as a linear kernel, and we denote the corresponding classifier Linear-SVM. Since the linear kernel has no parameters, the optimisation of Linear-SVM is controlled by a single slack parameter C, which "softens" the constraints to allow for noisy data (for a definition see, [70]). Another popular choice of kernel

function is a radial kernel function [71] (Radial-SVM), where the optimisation is controlled by two parameters. The first parameter C is the same as that in the linear kernel function. The second parameter γ is the inverse "width" of the radial kernel function, $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are two feature vectors. Function `svm` implements Linear-SVM and Radial-SVM, which is an interface to `libsvm` [72]. (Table 3.1).

One variation of the standard SVM can be derived by replacing the L2-norm regularizer in the objective function with an L1-norm regularizer [73] (L1-SVM). The optimisation problem can be formulated as

$$\min_{\mathbf{w},b} \quad \sum_{i=1}^{m} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)), \tag{3.10}$$

$$s.t. \quad ||\mathbf{w}||_1 \leq \lambda. \tag{3.11}$$

This produces sparse solutions where only a certain proportion of the feature variables are associated with non-zero weights and hence used to form the optimal hyperplane. This proportion is controlled by the parameter λ, where a smaller λ leads to fewer non-zero weights. Function `svm.fs` implements L1-SVM (Table 3.1).

As the SVM was originally designed for binary classification, the package we use solves multi-class problems using a one-vs-one scheme. It first trains a binary classifier for each pair of candidate classes, voting for one of the two. The final prediction is obtained by taking the majority vote of the binary classifiers.

3.2.4 Graph-based semi-supervised learning (SSL)

Graph-based SSL [74, 75, 76] is a semi-supervised learning technique based on the graphical structure of the data.

Given a dataset $X = \{\mathbf{x}_1, ..., \mathbf{x}_l, \mathbf{x}_{l+1}, ..., \mathbf{x}_n\}$ with n samples, of which only $l << n$ are labelled, i.e. each $\mathbf{x}_i, i = 1, ..., l$ has a label $y_i \in \{-1, 1\}$, but the remaining $i = l+1, ..., n$ are unlabelled, a graph can be constructed using the n samples, where each vertex represents a sample and the edge weight w_{ij} between $\mathbf{x}_i$ and $\mathbf{x}_j$ represents the similarity of the two instances. A larger weight corresponds to a higher

degree of similarity between the labels of the two vertices. One of the heuristics used to specify edge weights is to construct a k-nearest neighbour graph (k-NNG), where each vertex defines its k-nearest neighbour vertices by Euclidean distance. Two vertices are connected if one is among the other's k-nearest neighbours. If $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected, the edge weight w_{ij} is either the constant 1, in which case the graph is said to have binary edge weights or be unweighted, or a function of the distance such as the Radial Basis Function (RBF) $w_{ij} = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2})$. If $\mathbf{x}_i$ and $\mathbf{x}_j$ are not connected, w_{ij} is assigned zero. Empirically, k-NNG with small k tend to perform well [76].

Once the graph is constructed, learning involves assigning labels to unlabelled vertices $\mathbf{x}_i, i = l+1, ..., n$ such that disagreement of labels between neighbouring vertices is minimised while satisfying the constraint that the assignment for $\mathbf{x}_i, i = 1, ..., l$ match their true labels y_i. This is done by solving the following optimisation problem

$$\min_{f \in R} \quad \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2, \tag{3.12}$$

$$s.t. \quad f_i = y_i, i = 1, ..., l, \tag{3.13}$$

where $sign(f_i)$ will be the predicted label for $\mathbf{x}_i$. The matrix form of the above problem is

$$\min_{\mathbf{f} \in R^n} \quad \mathbf{f}' L \mathbf{f}, \tag{3.14}$$

$$s.t. \quad \mathbf{f}_l = \mathbf{y}_l, \tag{3.15}$$

where $\mathbf{f} = [\mathbf{f}_l, \mathbf{f}_u]'$, $sign(\mathbf{f}_l)$ contains the predicted labels for labelled samples, $sign(\mathbf{f}_u)$ contains the predicted labels for unlabelled samples, $\mathbf{y}_l = [y_1, ..., y_l]'$ contains the true labels for the labelled samples, $L = D - W$ is the Laplacian matrix, D is a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} w_{ij}$ and $W = (w_{ij})_{n \times n}$ is the adjacency matrix of the graph. The analytic solution to the optimisation problem can be derived by

finding the stationary point of the objective function, which is

$$\mathbf{f}_u = -L_{uu}^{-1} L_{ul} \mathbf{y}_l, \tag{3.16}$$

where L_{uu} and L_{ul} are sub-matrices of L when partitioned into labelled and unlabelled components

$$L = \begin{bmatrix} L_{ll} & L_{ul} \\ L_{lu} & L_{uu} \end{bmatrix}. \tag{3.17}$$

Normalisation of the algorithm can be achieved by ensuring the degree of each vertex is unity, which is done by replacing the Laplacian matrix L with the normalised version $L = I - D^{-1/2} W D^{-1/2}$, where I is the identity matrix. Similarly to the SVM, the technique was originally designed for binary classification. Here, we extend it to multi-class problems with the same one-vs-one scheme that is used for SVM, where a binary classifier is trained for each pair of classes and the final prediction is the majority vote from all participating binary classes. SSL is implemented in-house (Table 3.1).

3.2.5 Software

Here, we list the software packages and functions for the machine learning techniques used in our experiments.

Algorithm	**Function name**	**R 3.1.3 package [77]**
k-NN	`knn`	class [63]
RF	`randomForest`	randomForest [65]
Linear-/Radial-SVM	`svm`	e1071 [78]
L1-SVM	`svm.fs`	penalizedSVM [79]
Graph-based SSL	`graphSSL`	VirusTaxonomy [80]

Table 3.1: Software packages and functions for the machine learning techniques used.

3.3 Hierarchical classification

Non-hierarchical classification aims to predict the class for a given sample, whereas hierarchical classification aims to predict a set of hierarchically structured classes for each sample. Comprehensive reviews of hierarchical classification can be found

in [81, 82].

3.3.1 Characterisations

In a hierarchical classification problem, the label of a sample is a vector, where each element is a class at a specific level of the hierarchy that the sample belongs to, and different elements are related through a hierarchical structure. A hierarchical classification problem can be characterised by three aspects. The first is the graph of the hierarchical structure. Possible variations include the tree or directed acyclic graph (DAG) structure, which differs in the number of parents a class can have. The second is the number of labels a sample can have at one level. Possible variations are single or multiple label classification. The third is the labelling depth of a sample. Possible variations include mandatory or non-mandatory leaf node prediction, which differ in whether it is mandatory to label each sample with the leaf node classes. The hierarchical classification problem studied in this thesis involves tree, single label and mandatory leaf node prediction.

Hierarchical classifiers can be categorized into three types depending on how the hierarchical information is used: the flat classifier (bottom-up) approach, local classifier (top-down) approach and global classifier (big-bang) approach. The flat approach is the simplest solution to hierarchical classification, and is essentially a regular multi-class classifier at the deepest level plus a post-processing step that fills in the higher level labels. In contrast, the local and the global approaches explicitly incorporate the class hierarchy into the classification. The local approach is an ensemble method that combines multi-class non-hierarchical classifiers (base classifiers), whereas the global approach involves formulating a single optimisation problem taking into account the class hierarchy. In the next few sections, we will explain in detail the procedures for applying flat, local and global classifiers to classify samples into hierarchical classes.

3.3.2 Flat classifier

The flat classification approach, which is the simplest way to address hierarchical classification problems, consists of predicting classes only at the leaf nodes and

then filling in higher level labels during post-processing. For instance, to build a flat classifier for the class structure shown in Fig. 3.1, it trains one multi-class classifier that distinguishes among the four leaf nodes 1.1, 1.2, 2.1, and 2.2. During testing, it classifies a test sample into one of the leaf nodes. Since the structure is a tree, there is only one unique path from a given leaf to the root. Hence once the leaf class is known, its ancestors are by definition also obtained..

The learning and prediction stages of this approach are essentially the same as for non-hierarchical classifiers. However, it provides an indirect solution to hierarchical classification problems because when a leaf class is assigned to a sample, all its ancestor classes are also implicitly assigned as the class hierarchy is known a priori. This very simple approach has the disadvantage of having to build a classifier to discriminate among a large number of classes (all leaf classes) with a small number of samples per class, without exploiting information about parent-child class relationships present in the class hierarchy.

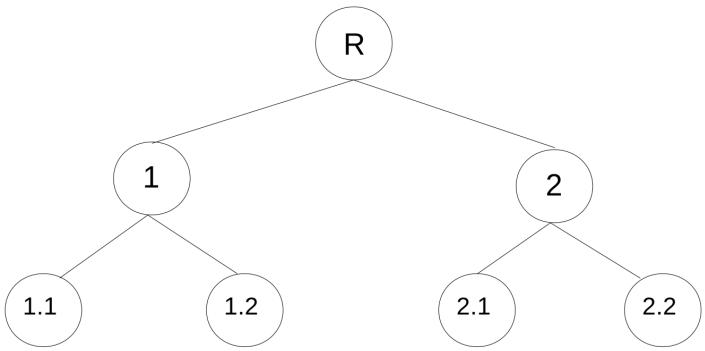


Figure 3.1: Class structure used to illustrate the procedures of applying different hierarchical classifiers.
The graph is a tree with seven nodes, each representing a class and the edges connecting them represent the parent-child relationship. The node R is the root class, sitting at the highest level of the hierarchy and is the ancestor of all other classes. Node 1 and 2 at the next level are two child classes of the root. Leaf nodes 1.1, 1.2, 2.1 and 2.2 are at the bottom of the hierarchy and are children of class 1 and class 2. The goal of the hierarchical classifier is to predict, for a sample, the set of classes forming a complete path from the root to the leaf.

3.3.3 Local classifier

The local classifier approach incorporates class hierarchy information by combining multiple base classifiers. Different methods differ in how they use local information and how they build their classifiers around it. Specifically, there are three standard ways of using the local information [81]. The first is one Local Classifier per Node (LCN), which trains base classifiers to distinguish a certain class among all the others. The second is one local classifier per level (LCL), which trains base classifiers to distinguish a certain class among all the others in the same hierarchical level. The third is one Local Classifier per Parent node (LCPN), which trains base classifiers to distinguish a certain class among all its siblings. We will illustrate the procedures for applying the three local classifiers using the class structure shown in Fig. 3.1.

LCN. When applying the LCN approach, six binary classifiers will be trained, one per node excluding the root: classifier 1 distinguishes node 1 from the rest, classifier 2 distinguishes node 2 from the rest, and so forth. During testing, a test sample would be classified into one of the two classes by each binary classifier, and the final predicted class path obtained after a post-processing step to correct any class inconsistency (see section "Inconsistency correction for LCN and LCL").

LCL. When applying the LCL approach, two multi-class classifiers will be trained, one per level excluding the root level: classifier 1 for nodes 1 and 2 at level 1, classifier 2 for nodes 1.1, 1.2, 2.1 and 2.2 at level 2. If the base classifier is originally designed for binary classification, such as SVM and graph-based SSL, we generalise them to cope with multi-class problems using a one-vs-one scheme (see section "SVM" and "RF" for details). During testing, a test sample is classified into one of the classes at the corresponding level by the two multi-class classifiers, and the final predicted class path is obtained after a post-processing step to correct any class inconsistency (see section "Inconsistency correction for LCN and LCL").

LCPN. When applying the LCPN approach, three multi-class classifiers will be trained, one per parent node: classifier 1 for nodes 1 and 2 whose parent is the root, classifier 2 for nodes 1.1 and 1.2 whose parent is node 1, and classifier 3 for nodes 2.1 and 2.2 whose parent is node 2. Similarly to the LCL approach, we adopt the

one-vs-one scheme to generalise the base binary classifier to multi-class problems. During testing, it first uses classifier 1 to classify a test sample into node 1 or node 2. It then decides whether to use classifier 2 or classifier 3 based on the outcome from classifier 1. If the sample is classified into node 1, it will use classifier 2 to classify it into either node 1.1 or 1.2; if the sample is classified into node 2, it will use classifier 3 to classify it into either node 2.1 or 2.2.

Inconsistency correction for LCN and LCL. A potential problem in the testing procedure of LCN and LCL is class inconsistency, where a sample is classified into classes at different levels, but these classes do not form a path from the ancestor class to the descendent class. That is, a class at the deeper level is not a descendant of a class at the higher level. Several solutions are discussed in [81]. For samples from the same level of our dataset, the difference between LCN and LCL is analogous to the difference between the one-vs-all and one-vs-one multi-class classification schemes. Hence, inconsistency correction in these cases requires a different approach. We adopt the method proposed in [83] for LCN and that in [1] for LCL.

For LCN, we correct inconsistency by choosing the path that gives the maximum probability. The probability of a path is computed as the product of the probability of a sample belonging to every class in that path. Platt [84] proposed to convert the SVM prediction of a sample $\mathbf{x}$ to the probability of the sample belonging to the predicted class i using

$$p_i(\mathbf{x}) = \frac{1}{1+e^{\alpha_i \mathbf{w}_i^T \mathbf{x}+\beta_i}}, \tag{3.18}$$

where $\mathbf{w}_i$ is the vector of SVM coefficients, and α_i and β_i are parameters. Computing probability of class membership in this way is shown to perform well in predicting posterior probabilities in practice [85], and also gives good performance in an SVM-based LCN approach [83]. In our SVM experiments, we replace the term $\alpha_i \mathbf{w}_i^T \mathbf{x} + \beta_i$ with the negative SVM output $-(\sum_{i=1}^{m} \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}) + \hat{b})$. For SSL, we use the same technique but replace the term with the negative SSL output $-f_i$.

For LCL, we adopt the strategies proposed in [1], which addresses the incon-

sistency problem via three scenarios. In scenario 1, there is a path P with a higher number of votes than the other paths and the class predicted at the lowest level in P is a leaf class. In this case, for each level N where the predicted class does not belong to the path P, the treatment of inconsistency replaces the predicted class at level N by the class in the path P belonging to the level N. Fig. 3.2 shows an example of this scenario. The predicted classes for each level are (Fig. 3.2 (a)): 2, 2.1, 3.2.1, and 2.1.2.1. The path with the highest number of votes contains the classes 2, 2.1, and 2.1.2.1, the last one being a leaf class. The predicted class in level 3 – 3.2.1 – does not belong to this path and the inconsistency elimination procedure transforms this class into the class 2.1.2 (Fig. 3.2 (b)).

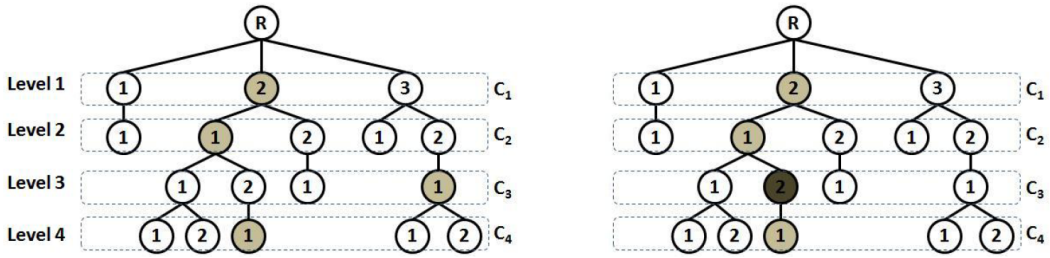
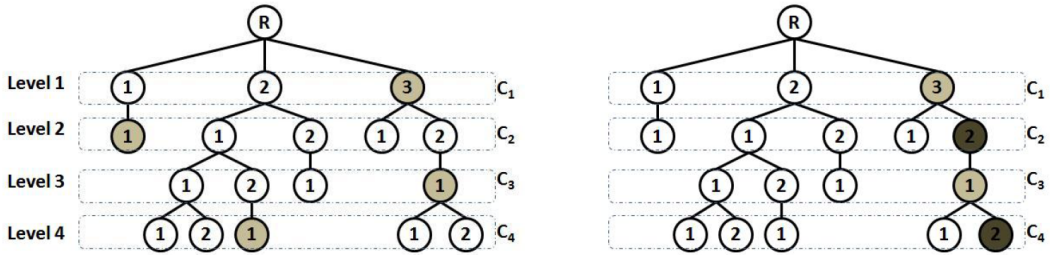


Figure 3.2: Illustration of class inconsistency for scenario 1 [1].

In scenario 2 there is a path P with a higher number of votes than the other paths but without a leaf predicted class in P. In this case, the class-inconsistency treatment consists of two steps: (1) for each level N higher than the lowest level that contains a predicted class in P, if the class in N does not belong to P, the predicted class in N is replaced by the class in the path P belonging to level N; (2) for the levels below the lowest level that contains a predicted class in P, the classes are determined using the LCPN strategy, i.e., for the class node with the lowest level in P, a classifier is built taking into account only its child classes. Then this classifier chooses one of the child classes, including it in P. If this chosen class is a leaf class, the treatment is finished. Otherwise the process is repeated for the next level, and so on, until a leaf class is included in P. Fig. 3.3 illustrates this scenario. The predicted classes for each level are (Fig. 3.3 (a)): 3, 1.1, 3.2.1, and 2.1.2.1. The path with the highest number of votes contains the classes 3 and 3.2.1, where level

3 is the lowest level with a predicted class in P. The strategy, as a first step, verifies that there is a predicted class in level 2 (1.1) which does not belong to this path and eliminates the inconsistency by replacing the predicted class in this level by the class 3.2 (Fig. 3.3 (b)). In the second step, the strategy applies the LCPN approach for the non-leaf class 3.2.1 by choosing one class between its children 3.2.1.1 and 3.2.1.2. The output class 3.2.1.2 is included in the path P, and, since it is a leaf class, the inconsistency treatment is concluded (Fig. 3.3 (b)).



(a) Inconsistent classification (b) Inconsistency elimination

Figure 3.3: Illustration of class inconsistency scenario 2 [1].

In scenario 3, there are multiple paths with the highest number of votes. In this case, the class inconsistency treatment initially chooses among these tied paths the path P that contains the predicted class at the highest level. After choosing the path P, if the predicted class of the lowest level in P is a leaf class, the procedure of Scenario 1 is used. Otherwise, the procedure of Scenario 2 is used. Fig. 3.4 presents an example of this third scenario. The predicted classes for each level are (Fig. 3.4 (a)): 3, 2.1, 2.1.2, and 3.2.1.2. There are two paths with two votes: one contains the predicted classes 3 and 3.2.1.2, and the other, the predicted classes 2.1 and 2.1.2. The system initially chooses the path that contains the predicted class 3, since this class is the nearest class to the root. After choosing this path, the example follows Scenario 1, because a leaf class belongs to this path. Hence, the class-inconsistency elimination procedure replaces the predicted classes 2.1 and 2.1.2, which do not belong to the chosen path, by the classes 3.2 and 3.2.1 (Fig. 3.4 (b)).

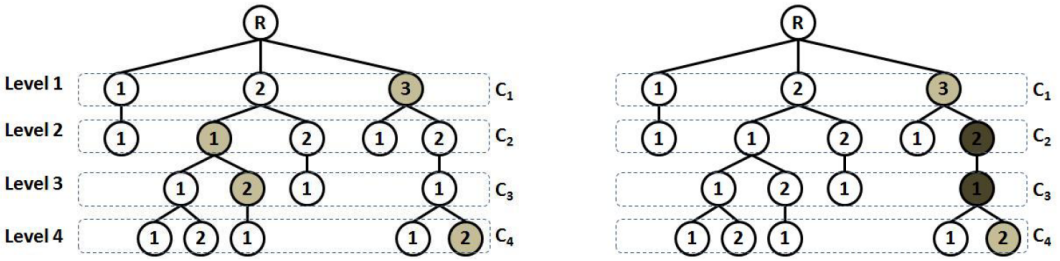


Figure 3.4: Illustration of class inconsistency scenario 3 [1].

3.3.4 Global classifier

Similarly to the local classifier approach, the global classifier (big-bang) approach considers class hierarchy during classification. However, it learns one single classifier for all classes explicitly taking the hierarchy into account. Its main distinction from the local classifier approach is the stage at which hierarchical information is incorporated. The global approach incorporates hierarchical information at the optimisation stage whereas the local approach incorporates it at the training and testing stage. The global approach is more complex and domain specific [81]. During training, it trains a single classifier for the entire tree, and during testing predicts a path from the root to the leaf for the test sample.

Studies show that approaches considering the class hierarchy are usually better than the flat classification approach, however, it is less clear if the global approach is better than the local approach [86, 87].

3.3.5 Relationship between hierarchical classifiers

The above categories of hierarchical classification are interrelated. When one-vs-all multi-class classification is used at individual levels, LCL is equivalent to LCN (see [81] for details of LCN and LCL); the inconsistency correction methods of LCL and LCN that assign zero weights to lower level predictions [88, 1] will produce the same results as LCPN, whereas those that prioritise the leaf level predictions will produce the same results as flat classifiers. The relationship between global classifiers is more complex since it depends on the base classifiers and the optimisation problem.

3.3.6 Bioinformatics applications

An important bioinformatics application of hierarchical classification is to gene expression and protein function prediction, which are not only hierarchical and multi-class but also multi-label [82]. There is no explicit study of hierarchical classification to classify viruses into ICTV taxonomic classes based on reference genome sequences. Related works, including [52, 48, 89] typically use flat classification for individual ICTV taxonomic levels without accounting for hierarchy.

3.4 Performance measures

To assess the performance of a learning algorithm on a problem, a suitable measure is important to correctly quantify the suitability of the algorithm to the given problem. Here, we describe performance measures used in non-hierarchical and hierarchical classification problems.

3.4.1 Non-hierarchical measures

The performance of a classification method can be measured in various ways (see [90] for a detailed discussion), with error rate (the ratio of the number of misclassified samples to the total number of samples) or accuracy (1 - error rate) being the most commonly used in similar studies [52, 48], which summarise the overall performance with a single value.

Detailed performance on individual classes can be broken down using a confusion matrix, which is a specific table layout. Each column of the matrix represents the samples in a true class while each row represents the samples in a predicted class (or vice versa). The name stems from the ease with which the table shows whether the system is confusing two classes (i.e. commonly mislabelling one as another).

In a binary confusion matrix, the following measures can be derived (Table 3.2): true positive (TP), the number of samples from a certain class being correctly assigned to that class; true negative (TN), the number of samples not from a certain class being correctly predicted as not in that class; false negative (FN), the number of samples from a certain class that failed to be assigned to that class; and false positive (FP), the number of samples not from a certain class but have been incor-

rectly assigned to that class. Hence, accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$ and error rate = 1 - accuracy.

For multi-class classification problems, the matrix can be generalised to include all classes (Table 3.3). The diagonal entries are correct classifications and off-diagonals misclassifications. Hence accuracy can be computed as the ratio of the trace of the matrix to the total sum of all its values.

	True +1	True -1
Predicted +1	TP	FP
Predicted -1	FN	TN

Table 3.2: Confusion matrix for binary classification of classes labelled with +1 or -1.

	True C_1	True C_2	...	True C_n
Predicted C_1	Correct	Incorrect	...	Incorrect
Predicted C_2	Incorrect	Correct	...	Incorrect
...	...	...	...	...
Predicted C_n	Incorrect	Incorrect	...	Correct

Table 3.3: Confusion matrix for multi-class classification of classes labelled with C_1, ..., C_n.

3.4.2 Hierarchical measures

One way to measure the performance of a hierarchical classifier is to measure its performance at individual levels. However, this can be cumbersome when the number of levels is large. In addition, classes at higher levels are more general and important, but those at deeper levels are more specific and difficult to train. It is useful to summarise the performance using a single value that accounts for the relationship between classes and measures the entire hierarchical problem as a whole.

Performance measures for hierarchical classifiers can be obtained by modifying their non-hierarchical counterparts by explicitly considering the relationships between classes (see [91, 90, 92] for reviews). The measures can be divided into

four types: 1) distance-based, which incorporates the distance between the predicted and true classes in the hierarchy and penalises greater distances more; 2) depth-based, which incorporates both class distance and depth into measures and penalises more for misclassification at higher levels; 3) semantics-based, which considers the semantic meaning or intrinsic nature of classes and penalises more for distinct ones; and 4) hierarchy-based, which explores the degree of overlap of ancestors and/or descendants between the predicted and true classes and penalises more for less overlap.

One type of distance-based measure is symmetric difference loss, defined as

$$l_{\Delta}(\mathbf{y},\hat{\mathbf{y}}) = \sum_{j} I(y_j \neq \hat{y}_j), \tag{3.19}$$

where $\mathbf{y}$ and $\hat{\mathbf{y}}$ are vectors of the true and predicted hierarchical labels, $I(.)$ is an identity function that equals one when the condition evaluates to true and zero if false. This loss accounts for the distance between the true and the predicted classes but ignores the depth of the classes.

The H-loss [88], short for Hierarchical-loss, is defined as

$$l_{H}(\mathbf{y},\hat{\mathbf{y}}) = \sum_{j} c_j I(y_j \neq \hat{y}_j \wedge y_i = \hat{y}_i, \forall i \in anc(j)), \tag{3.20}$$

where c_j is the coefficient associated with node j, and $i \in anc(j)$ means that i is an ancestor of j. The coefficients $0 \leq c_j \leq 1$ are used for down-scaling the loss when going deeper into the hierarchy. Different choices of c_j lead to different types of measures. We review some of the choices that have been used in [93], representing depth-based, semantics-based and hierarchy-based measures. The choice of a uniform weighting ($c_j = 1$) corresponds to a depth-based measure and is equivalent to error rates at Genus level in our problem. We denote it as l_{unif}. The loss can also be a semantics-based measure when the weights reflect the properties of branching conditions, such as by defining

$$c_{root} = 1, c_j = \frac{c_{pa(j)}}{|sibl(j)|}, j \neq root, \tag{3.21}$$

where $pa(j)$ is the immediate parent and $sibl(j)$ is the set of siblings of node j (including j itself). We denote this loss as l_{sibl}. A hierarchy-based measure can be derived by scaling the loss by the proportion of the hierarchy that is in the subtree $T(j)$ rooted by j, that is, to define

$$c_j = \frac{|T(j)|}{|T(root)|}. \tag{3.22}$$

This is denoted as l_{subtr}.

Chapter 4

Features for genome sequences

Since genome sequences do not explicitly possess features, we must first choose suitable features and extract them from the sequence. The transformation of long sequences of varying lengths into much shorter fixed length vectors is important to feature-based learning algorithms. This chapter describes the features for genome sequences used in this thesis.

4.1 Nucleotide-based features

Nucleotide-based features are vectors that consist of nucleotide-related summary statistics such as count, mean and variance. Genome length is the simplest statistic of the sequence and has been shown to distinguish different organisms [94, 95]. Another statistic, the GC content (the proportion of nucleotides G and C in the entire sequence), is also found to vary with different organisms or genes [96, 97, 94]. This is due to the variation in selection, mutational bias, and biased recombination-associated DNA repair [98]. In the following sections, we will review the Natural Vector representation, a recently proposed feature derived from simple nucleotide statistics.

4.1.1 Natural Vector

The Natural Vector (NV) methodology described in a recent work [39] is a one-to-one mapping that recasts a variable length nucleic acid sequence as a fixed-length low dimensional vector of global composition- and location-based statistics computed from the sequence. Given a genome sequence, the NV methodology com-

putes simple and intuitive global statistics for each nucleotide – count, mean position in the sequence, variance of the position and higher order moments. Let $S = s_1 s_2 \cdots, s_n$ be a sequence of length n where $s_i \in \{A,C,G,T\}, i = 1,2,\cdots,n$. For each nucleotide s, the summary statistics are the count n_s, mean position in the sequence μ_s and normalised variance of the position d_s. These are defined as

$$n_s = \sum_{i=1}^{n} I(s = s_i), \tag{4.1}$$

$$\mu_s = \frac{1}{n_s} \sum_{i=1}^{n} i\, I(s = s_i), \tag{4.2}$$

$$d_s = \frac{1}{n_s n} \sum_{i=1}^{n} (i - \mu_s)^2 I(s = s_i). \tag{4.3}$$

where $I(s = s_i)$ is the indicator function which equals 1 when $s = s_i$ and 0 otherwise.

Hence the 12-dimensional NV representation of a nucleic acid sequence S used in earlier work [52] is

$$NV12 = [\mathbf{n}, \boldsymbol{\mu}, \mathbf{d}], \tag{4.4}$$

where $\mathbf{n}$, $\boldsymbol{\mu}$ and $\mathbf{d}$ are the following vectors

$$\mathbf{n} = [n_A, n_C, n_G, n_T], \tag{4.5}$$

$$\boldsymbol{\mu} = [\mu_A, \mu_C, \mu_G, \mu_T], \tag{4.6}$$

$$\mathbf{d} = [d_A, d_C, d_G, d_T]. \tag{4.7}$$

For the task of predicting the Baltimore Class and ICTV Order of a non-segmented virus, good classification performance can be achieved using a 12-dimensional NV embedding of its genome sequence consisting of counts, mean positions and normalised variances (NV12) and the k-NN classifier; the inclusion of higher order moments does not improve results [52]. The NV12 representation performs well when used to classify the genome sequence of a multi-segmented virus and to determine phylogenetic relationships [99].

4.1.2 Natural Vector and its derivatives

In addition to the NV12, we also use subsets of the variables to create three other "simpler" features. Table 4.1 summarises the genome sequence features based on nucleotide statistics we examined.

Name	Dimensions	Variable(s)
Length	1	n
NV4	4	n_A, n_C, n_G, n_T
NV8	8	$n_A, n_C, n_G, n_T, \mu_A, \mu_C, \mu_G, \mu_T$
NV12	12	$n_A, n_C, n_G, n_T, \mu_A, \mu_C, \mu_G, \mu_T, d_A, d_C, d_G, d_T$

Table 4.1: Genome sequence features based on nucleotide statistics.

The name, dimensionality, and components of each feature vector are given. The summary statistics used in the representation of a sequence are the count n_s, mean position in the sequence μ_s, and normalised variance of the position d_s of a nucleotide $s \in \{A, C, G, T\}$.

4.2 Word-based features

A word, also called a k-mer, is a short sequence of consecutive k nucleotides $w = s_i s_{i+1} \ldots s_{i+k-1}$. They can be extracted from the sequence by sliding a window of length k over the genome from the first position to the $(n-k+1)$-th position. With an alphabet consisting of a nucleotides, there are a^k possible k-mers that can appear in a genome sequence.

4.2.1 k-mer count

The first and most developed alignment-free methods use the frequency of k-mers for genome sequence comparison [37]. In their work, the authors modelled sequences as Markov chains and quantified the difference between two sequences by the squared Euclidean distance between their transition matrices. Since then, methods based on the frequency or count of k-mers have significantly advanced, with the introduction of more sophisticated sequence models [100, 16], effective distance measures [11], as well as the accommodation of mismatched characters in the words [101]. The methods have also been applied to numerous tasks relating to biological sequences, such as comparing sequence similarity [37, 102, 103, 104, 105],

identifying enhancer [106] and microRNA [107], explaining variability in mutation rates of the human genome [108, 109], as well as assembling short reads generated from next-generation sequencing [110, 111].

4.2.2 k-mer Natural Vector

A potential deficiency of count is that the positional information of k-mers within a sequence is neglected, and hence the inter-relationships between different k-mers are lost. In contrast, the NV of a genome incorporates positional information to account for the inter-relationships between different portions of genetic sequences. However, it neglects information from short nucleotide sub-sequences in the genome.

Recent works exploit the complementary nature of the k-mer model and the NV. The authors of [40] propose to incorporate k-mers into NV by generalising $\mathbf{n}$, $\boldsymbol{\mu}$ and $\mathbf{d}$ to the count, mean position and normalised variance of all possible k-mers:

$$\mathbf{n} = [n_{w_1}, n_{w_2}, ..., n_{w_{a^k}}], \tag{4.8}$$

$$\boldsymbol{\mu} = [\mu_{w_1}, \mu_{w_2}, ..., \mu_{w_{a^k}}], \tag{4.9}$$

$$\mathbf{d} = [d_{w_1}, d_{w_2}, ..., d_{w_{a^k}}]. \tag{4.10}$$

Therefore, a k-mer NV with an alphabet of size a has $3 \times a^k$ variables in total. For k-mers that are absent in a sequence, their counts, mean positions and normalised variances are set to zero. Similarly to the NV, k-mer NV preserves the one-to-one relationship with the original genome sequence. In the special case where $k = 1$ and the alphabet is $\{A, C, G, T\}$, 1-mer NV is the 12-dimensional NV (NV12). Experimental results show improved accuracy in revealing evolutionary relationships of species considered in phylogenetic analysis of genetic sequences compared to NV.

A number of similar works exist. The authors in [89] found that using a difference measure combining k-mer counts with NV improves performance in classification using the 1-NN classifier and phylogenetic tree analysis [89]. The method in [89] differs from that in [40] in the sense that the former concatenates k-mer counts with NV, whereas the latter is k-mer NV, which includes the count, mean position

and normalised variance of k-mers in a genome sequence. Another work [48] proposes the Generalized Vector (GV), which concatenates the Complete Composite Vector (CCV) with the k-mer NV.

4.2.3 Return time distribution

Another representation that accounts for both the nucleotide sequence composition and their relative orders is the Return Time Distribution (RTD) [112, 113, 114]. The Return Time (RT) of a k-mer is defined as the number of nucleotides between the successive appearances of the k-mer, and the frequency distribution of those RT is referred to as a RTD, which includes the mean and standard deviation of the RT of that k-mer. Hence, the feature vector representation of a sequence consists of the mean and standard deviation of the RT of all possible k-mers in the sequence.

Let $p^t_{w_i}$ denote the position of the t-th occurrence of k-mer w_i in a sequence, and if it occurs T times in total, then the t-th return time is

$$\Delta p^t_{w_i} = p^t_{w_i} - p^{t-1}_{w_i}, 2 \leq t \leq T. \tag{4.11}$$

The RTD, summarised by the mean and standard deviation of the series of return times are

$$\mu_{\Delta p^t_{w_i}} = \frac{1}{T}\sum_{t=2}^{T} \Delta p^t_{w_i}, T \geq 2, \tag{4.12}$$

$$d_{\Delta p^t_{w_i}} = \frac{1}{T}\sum_{t=2}^{T} (\Delta p^t_{w_i} - \mu_{\Delta p^t_{w_i}})^2, T \geq 3. \tag{4.13}$$

For k-mers that occur twice ($T = 2$), there is only one return time and hence we set the standard deviation to zero; for those that occur less than twice ($T < 2$), there is no return time and for convenience, we set both the mean and standard deviation to zero.

The method was evaluated using simulated data and successfully applied to the molecular phylogenetic analysis of the virus family *Flaviviridae*, and subtyping of *Dengue viruses* [113], as well as the genotyping of *Mumps viruses* based on sequences of small hydrophobic genes [115].

4.2.4 Summary of features

In addition to the standard nucleotide alphabet $\{A, C, G, T\}$ (ACGT), we also investigated the performance when using other alternative alphabets (Table 4.2). Table 4.3 summarises feature representations involving k-mer statistics used in our study.

Alphabet	**Size**	**Element**	**Description**	**Bases**
ACGT	4	A	Adenine	A
		C	Cytosine	C
		G	Guanine	G
		T	Thymine	T
SW	2	S	Strong	C, G
		W	Weak	A, T
RY	2	R	puRine	A, G
		Y	pYrimidine	C, T
MK	2	M	aMino	A, C
		K	Keto	G, T

Table 4.2: Alphabets used to construct feature vectors.

Feature	Alphabet	***k*-mer**					
		1-mer	2-mer	3-mer	4-mer	5-mer	6-mer
Counts	SW	2	4	8	16	32	64
Counts	RY	2	4	8	16	32	64
Counts	MK	2	4	8	16	32	64
Concatenated counts	SW, RY, MK	6	12	24	48	96	192
Counts	ACGT	4	16	64	256	1,024	4,096
RTD	ACGT	8	32	128	512	2,048	8,192
Counts, RTD	ACGT	12	48	192	768	3,072	12,288
k-mer NV	ACGT	12	48	192	768	3,072	12,288
Concatenated k-mer NV	ACGT	12	60	252	1,020	4,092	16,380

Table 4.3: Dimensions of different k-mer feature vectors.

Table shows the dimension of each type of k-mer based features given a particular alphabet set and value of k. Concatenated k-mer NV is represented by concatenating NV of k-mers and all their subwords.

4.2.5 Relationship to string kernels

The marriage between features derived from word statistics and SVMs is closely related to string kernel methods, which involve first constructing a kernel to quantify the similarity between sequences and then classifying them into different classes using kernel-based methods. The k-spectrum kernel method proposed in [116] is equivalent to the combination of k-mer counts and Linear-SVM in our work. The kernel accounts for the number of occurrences but not positional information. In contrast, the weighted degree kernel [117] considers positional information of k-mers but ignores their occurrences in individual sequences. They count the co-occurrences of the same k-mers at corresponding positions in the two sequences to be compared, and then compute a sum weighted by coefficients related to the value of k. The method is related to the combination of concatenated k-mer NV and Linear-SVM in our work.

4.2.6 Absent words

The features described above represent a genome sequence using its composing elements. This section describes an opposite approach which exploits the representational power of words that are possible but absent in the sequence. A pioneering work [42] defines the term "nullomer" to be the shortest words that do not occur in a given genome and the term "prime" to be the shortest words that are absent from the entire known genetic data. Their motivation was to discover the constraints on natural DNA and protein sequences.

Minimum absent words (MAW). According to [43], MAW are words that are themselves absent in a sequence but any subwords of which are present. The set of MAW has favourable properties because it includes the shortest absent words, but is also much easier to manipulate than the set of absent words. The paper illustrates that as the length of the k-mer increases (i.e. k increases), the number of absent words grows exponentially, whereas the number of MAW grows until a maximum and then decreases towards zero. A later work [118] also shows that the characteristic distribution of genomic MAW as a function of their length are qualitatively similar for all living organisms in the study, with the majority being rather

short. The paper also demonstrates that long MAW are of biological origin and can be used as markers for genomic complexity. Other works such as [119, 120] have used MAW for inter- and intra-species comparisons.

Difference measures. A number of difference measures for the MAW of sequences are studied in [121]. As they found Length-Weighted Index (LWI) with set intersection and Jaccard Distance (JD) to be the best, they will be used in our study:

$$LWI_{\cap}(S_1,S_2) = -\sum_{w \in MAW_{S_1} \cap MAW_{S_2}} \frac{1}{|w|^2}, \tag{4.14}$$

$$JD(S_1,S_2) = 1 - \frac{|MAW_{S_1} \cap MAW_{S_2}|}{|MAW_{S_1} \cup MAW_{S_2}|}. \tag{4.15}$$

where S_1 and S_2 are two sequences, and MAW_{S_1} and MAW_{S_2} are the sets of their MAW respectively.

4.3 Compression-based features

The above methods treat a genome as an ordered sequence of words and construct features using statistics of present or absent words. Here, we treat a genome as a piece of text and construct features using the compressibility of information contained in a sequence, with the assumption that similar sequences contain similar patterns and hence result in similar compression ratios.

The use of savings in joint compression as a measure of similarity is founded on information theory and coding [41]. The fundamental concept behind the distance metric proposed is that of Kolmogorov complexity [122], which is a measure of the computational resources needed to specify an object. It reflects the complexity of the underlying structure of the object and is defined as the length of the shortest computer program (in a predetermined programming language) that produces the object. There are presently no absolute measures of Kolmogorov complexity and it can only be estimated using compression algorithms that are assumed to be efficient [123]. The compression ratio, defined as the ratio between the compressed file size to the original file size, reflects the complexity of the file content.

This section reviews a number of high-performing commonly used compres-

sions tools, both general-purpose and DNA-specific.

4.3.1 General compression

This section describes the efficient and popular general purpose compression tools used in our study.

gzip. gzip compresses the size of a given file using Lempel-Ziv coding (LZ77). Whenever possible, each file is replaced by one with the extension .gz. The Lempel-Ziv compression algorithm [124] is a dictionary compression scheme that works by finding duplicated strings in the input text, and replacing these duplicates with a reference (distance and length) to the original string. The distances are compressed with one Huffman tree, and the lengths with another. The gzip implementation limits its distances to 32K bytes, and lengths to 258 bytes. Furthermore, implementation heuristics are used to try to improve performance and avoid worst-case scenarios. For instance, lazy match evaluation is used to try to select the longer matches when multiple matches appear together.

bzip2. bzip2 compresses files using the Burrows-Wheeler block sorting text compression algorithm [125], along with Huffman coding [126]. Whenever possible, each file is replaced by one with the extension .bz2. The Burrows-Wheeler compression algorithm applies a reversible transformation to a block of input text with the aim of making the transformed text easier to compress with simple algorithms. For an n-character string s, the transformation forms the n rotations (cyclic shifts) of s, sorting them lexicographically and extracting the last character of each rotation into a list. This list is the transformed character string. This transformation tends to bring the same characters closer together, which is easy to compress with simpler algorithms such as move-to-the-front and Huffman coding. In general the algorithm is slightly slower than that of Lempel-Ziv (as used by gzip), but achieves better compression levels.

xz. xz is relatively new compared to the others and uses the Lempel-Ziv-Markov chain (LZMA) compression algorithm [127]. LZMA uses a variant of Lempel-Ziv coding (LZ77) with large dictionary sizes and support for repeatedly used match distances, with the output encoded with a range encoder. Range encoding is an

entropy coding method, similar to arithmetic coding, except with digits expressed in any base as opposed to just bits (base 2). This has several benefits over Huffman encoding, for instance a better than 1-bit per symbol compression lower-bound.

zip. zip is a compression and file packaging utility for Unix/Linux and each file is stored in single file with the extension .zip. The zip file format supports lossless data compression via a number of compression algorithms. By default (and in our usage) zip uses the DEFLATE compression algorithm, which itself uses a combination of Lempel-Ziv (LZ77) and Huffman coding and is described in [128].

4.3.2 DNA specific compression

There exist numerous DNA specific compression methods; comprehensive reviews can be found in [129, 130, 131, 132]. These methods can be divided into two categories: reference-based approaches, which compress DNA sequences based on their relative difference from a set of reference sequences, and reference-free approaches, which compress sequences without referring to others but by exploring compressible patterns within them. The difference between reference-based and reference-free approaches is analogous to the difference between alignment-based and alignment-free sequence comparison methods.

Reference-based and reference-free approaches lead to different forms of the classification problem. When a reference-free approach is used, each genome sequence file can be compressed independently and the compression quality can be quantified using a single number – the compression ratio. With multiple reference-free methods, a number of compression ratios can be collected and a feature vector for a sequence file can be constructed by concatenating those ratios, which can then be fed into classifiers. The existence of feature vectors enables the application of both feature and distance based classifiers.

In contrast, when reference-based approaches are used, the compression ratio of a file is dependent on the reference file used, i.e. a different reference file leads to a different compression ratio for the same file. Therefore, the compression quality of a reference-based approach is more suitable to being expressed as a matrix consisting of pairwise relative compression ratios, which can be obtained by run-

ning compression on each pair of sequence files in the dataset. Without a feature vector, only distance based classifiers can be applied. We will construct features using compression ratios of reference-free methods, since they allow for more of our classifiers to be applied.

Compression methods can also be specialised for certain file formats. We will describe reference-free methods specialised for FASTA files, which are used in our dataset.

DELIMINATE. DELIMINATE [133] (a combination of Delta encoding and progressive ELIMINATion of nucleotide characters) is a fast and efficient method for lossless compression of genomic sequences. It performs compression of sequence data in two phases. In the first phase, information of all non-ACGT characters and low complexity regions (represented by lower case characters) are recorded. The file is then stripped of non-ACGT characters, and all characters are converted to upper case. The resulting file, which contains only four distinct characters is processed in the second phase. In this phase, the positions of two nucleotides having the highest frequencies of occurrence are delta encoded, which stores the differences rather than the original values, and these bases are subsequently eliminated from the sequence. The remaining two bases (having the least frequencies) are then represented with a binary code. It outperforms general-purpose tools and the compression gains with large sequence datasets are dramatically higher.

MFCompress. MFCompress [134] (Multi-fasta and Fasta Compression) is another lossless method that is claimed to provide better compression than DELIMINATE for the large majority of the files used in the benchmarking dataset, for a similar compression and decompression time. MFCompress relies on multiple competing finite-context models and arithmetic coding [135]. The finite-context models used are probabilistic models that comply with the Markov property, and use the immediate past symbols to select the probability distribution of the next one. One single finite-context model is used for encoding the header, and multiple context models for the main stream of the DNA sequences. The compression algorithm divides the data source into two separate sub-sources: one containing the headers of the FASTA

records, the other one the sequences. The sub-source that deals with the sequences may be further divided into three streams: the main stream, the extra stream and the case stream. The main stream is a four-symbol information source, conveying most of the information of the four DNA bases (ACGT). Both upper and lower case characters representing the four DNA bases are converted to this four-symbol alphabet. If characters other than the four DNA bases are also present, they are all mapped to the "0" symbol in the main stream. The extra stream is responsible for representing all non-ACGT characters, both in upper and lower case, that have been found in the sequences, as well as to indicate when the "0" in the main stream is an "A" DNA base. When the sequences contain both DNA bases in upper and lower case, the case stream is used, which uses an additional binary symbol to indicate the case type of a DNA base in the main stream.

LEON. LEON [136] is a method designed to compress data issued from high throughput sequencing techniques. The method is based on a reference probabilistic de Bruijn Graph, built from the set of short sequences (called reads) acquired by a sequencing device and stored in a Bloom filter [137], a space-efficient probabilistic data structure used to test whether an element is a member of a set. Each read is encoded as a path in this graph, by memorising an anchoring k-mer and a list of bifurcations, enough to rebuild it from the graph. The same probabilistic de Bruijn Graph is used to perform a lossy transformation of the quality scores, which allows for higher compression rates without losing pertinent information for downstream analysis. Compression of sequencing data can be divided into three distinct problems: compression of read IDs, of base sequence and of quality scores. For the compression of read IDs, standard methods are perfectly suited, since read IDs are usually highly similar from one read to another. Compression of DNA sequences and quality scores on the other hand are two very different problems. The former displays high redundancy across reads when the depth of sequencing is high but spread over the whole file, and must be lossless, whereas the latter displays a highly noisy signal on a larger alphabet size, in which case lossy compression may be appropriate.

4.3.3 Summary of features

The features for each virus genome sequence are constructed by concatenating the compression ratios of different tools and are summarised in Table 4.4.

Name	Feature	Dimension	Variables
CRGP	Compression Ratio of General Purpose tools	4	$r_{bzip2}, r_{gip}, r_{xz}, r_{zip}$
CRB	Compression Ratio of Bzip2	1	r_{bzip2}
CRBL	Compression Ratio of Bzip2 and log Length	2	$r_{bzip2}, log(Length)$
CRDNA	Compression Ratio of DNA specific tools	3	$r_{DELIMINATE}, r_{MFCompress}, r_{LEON}$
CRL	Compression Ratio of LEON	1	r_{LEON}
CRLL	Compression Ratio of LEON and log Length	2	$r_{LEON}, log(Length)$
CRA	Compression Ratio of All	7	$r_{bzip2}, r_{gip}, r_{xz}, r_{zip}$ $r_{DELIMINATE}, r_{MFCompress}, r_{LEON}$
CRLB	Compression Ratio of LEON and Bzip2	2	r_{LEON}, r_{bzip2}

Table 4.4: Features based on compression ratios.

Table shows the name (abbreviation) of the features used, the dimension and variables of the features. r_{tool} represents the compression ratio of the tool.

Chapter 5

Exploratory data analysis

As genome sequences are high-dimensional data, visualising them and understanding their properties is a non-trivial task. In this chapter, we conduct an exploratory analysis of the virus genome sequence dataset using various visualisation techniques and emphasise statistical properties that contribute to the separability of sequences from different taxonomic classes. Our aim is to assist hypothesis formulation and experimental design, and provide intuition and a qualitative understanding of the experimental results in later chapters.

5.1 Methods

Different plots can visualise different aspects of the dataset. A box plot is a concise way to summarise key statistics of a dataset, a ternary plot shows the relationship between three variables in a two-dimensional space, and a t-SNE plot is specialised in displaying very high dimensional data in a low dimensional space.

5.1.1 Box plots

A box plot, or box-and-whisker plot is a convenient way of graphically depicting groups of numerical data through their descriptive statistics (Fig. 5.1). The plot displays the inliers and outliers of a dataset using three main components: box, whisker and circle. The bottom, internal band, and top of the box are the first, second (median), and third quartiles of the data respectively; the spacings between the different parts of the box indicate the degree of dispersion and skewness in the data. The bottom and top whiskers are the minimum and the maximum values of inliers.

Each circle represents an outlier, a data point whose value is more than 1.5 times the inter-quartile range. Box plots are non-parametric, meaning that they display the variation in samples of a statistical population without making any assumptions about the underlying statistical distribution. Box plots may seem more primitive than a histogram but they take up less space and are therefore particularly useful for comparing distributions between several groups or sets of data.

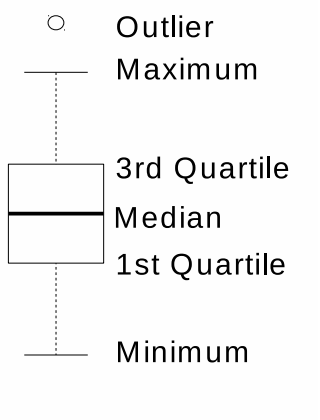


Figure 5.1: Anatomy of a box plot.

5.1.2 Ternary plots

A ternary plot, also known as a simplex plot or de Finetti diagram, uses a triangular coordinate system to display three variables in a two-dimensional space (see Fig. 5.14 left subplot). The coordinate system is organised as an equilateral triangle, whose vertices are associated with three variables. The three variables define the position of a point in the triangular coordinate system, which represents the location of the barycentre of the triangle when each vertex has weight equal to the value of the corresponding variable. The distance from a point to a vertex is inversely proportional to the magnitude of the value of the variable associated with the vertex, as opposed to that in a Cartesian coordinate system. A point at a vertex of the triangle has non-zero value for the corresponding variable but zero for the other two; a point at a median of the triangle has equal value for the other two variables; a point at an edge of the triangle has zero value for the variable of the corresponding vertex but non-zero for the other two. The advantage of using a ternary plot for

depicting compositions is that three variables can be conveniently plotted in a two-dimensional graph.

5.1.3 t-SNE plots

t-distributed Stochastic Neighbour Embedding (t-SNE) is an unsupervised nonlinear dimensionality reduction technique [138]. It models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modelled by nearby points and dissimilar objects are modelled by distant points. The t-SNE algorithm comprises two main stages. First, it constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked, whilst dissimilar points have an extremely small probability of being picked. Second, it defines a similar probability distribution over the points in the low-dimensional map, and minimizes the Kullback-Leibler divergence between the two distributions with respect to the locations of the points in the map. It is particularly well-suited for embedding high-dimensional data into a space of two or three dimensions, which can then be visualized using a regular scatter plot.

5.1.4 Software

Table 5.1 summarizes the software packages and functions used for visualisation in our studies.

Type	**Function name**	**R package**
Box plot	`boxplot`	graphics [139]
Ternary plot	`ggtern`	ggtern [140]
t-SNE plot	`Rtsne`	Rtsne [141]

Table 5.1: Software packages and functions used for visualisation

5.2 Visualisation of the dataset

We investigate the properties of the virus genome sequence dataset in terms of nucleotide statistics, and word- and compression-based feature representations. We focus on non-satellite and non-segmented virus genome sequences in the dataset.

5.2.1 Features based on nucleotide statistics

In this section, we analyse the distribution of nucleotide statistics of the genome sequences in the dataset. Fig. 5.2 summarises the distributions of Length, a simple single-value feature, for individual classes. They appear distinguishable between different classes; ICTV Orders in particular and Baltimore Classes to a lesser extent.

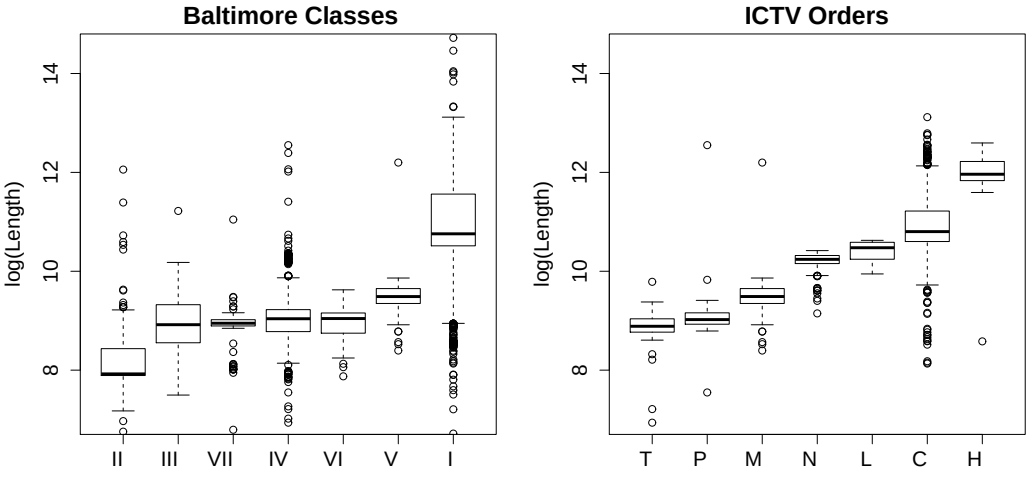


Figure 5.2: Box plots of Length for each class of the Baltimore and ICTV Order schemes.
Since there is a large dynamic range in Length, we display the natural log of the values in one plot. Classes are ordered by increasing median Length.

The distributions of slightly more sophisticated statistics including nucleotide count, mean position in the sequence and normalised variance of the position appear identical to that of Length but smaller in magnitude (Fig. 5.3 - 5.4). Histograms of these statistics drawn from all the sequences in the dataset show that the distributions of these global composition- and location-related variables are not uni-modal (Fig. 5.5 and 5.6). Instead, they have a multi-modal distribution, often with four spikes for Baltimore Classes and three for ICTV Orders. When the mean nucleotide position is normalised by genome length, the value peaks at around half (Fig. 5.7). Scatter plots between the statistics reveal a strong linear correlation, with the correlation between the mean position and normalised variance being the strongest (Fig. 5.8).

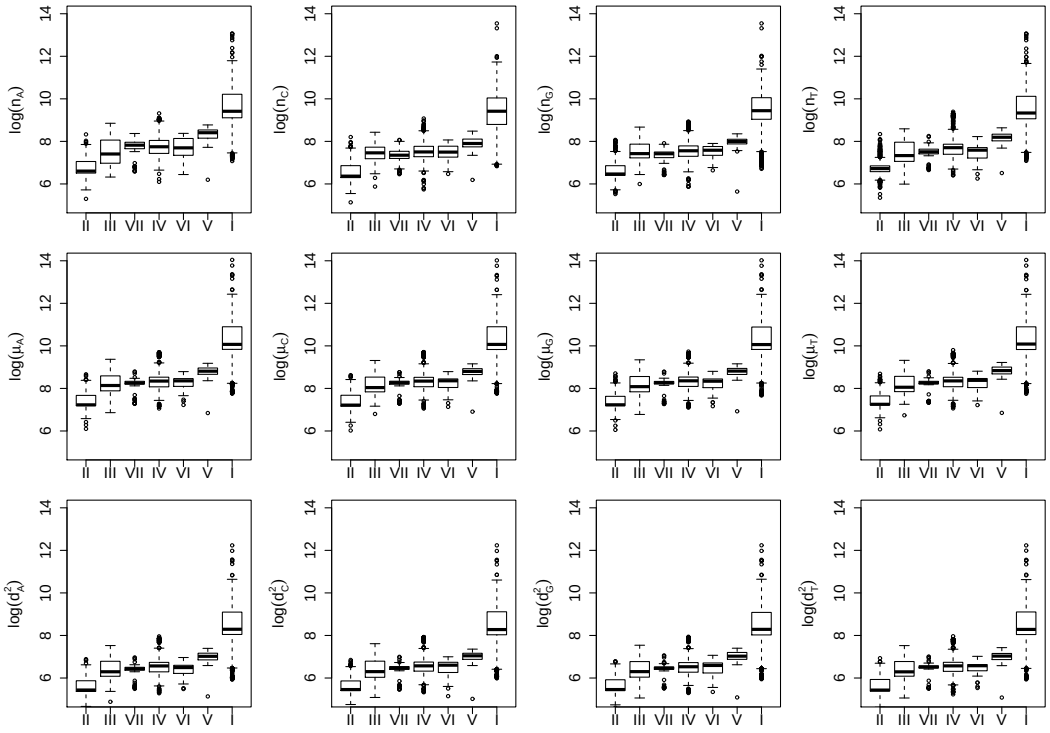


Figure 5.3: Box plots of summary statistics for genome sequences with Baltimore Class labels.

The statistics depicted are nucleotide count (n_A, n_C, n_G, n_T), mean nucleotide position in the sequence $(\mu_A, \mu_C, \mu_G, \mu_T)$ and normalised variance of the position $(d_A^2, d_C^2, d_G^2, d_T^2)$. Each box plot is a visual summary of the inliers and outliers in the data set for a particular taxonomic category, the natural log transformed values of a statistic for viruses assigned to a specific Baltimore Class. In each panel, the Baltimore Classes are ordered by increasing median Length.

We compute coordinates of viruses using t-SNE with a random seed and NV4, NV8 and NV12 as inputs. The embedding is produced in an unsupervised setting as similar sequences are automatically assigned nearby coordinates. Colour coded scatter plots reveal interesting clusters in the dataset (Fig. 5.9). The results show that viruses from the same classes form clusters and those from different classes can be properly separated. In addition, classes with similar Length distributions in Fig. 5.2 are neighbours in the t-SNE plots.This becomes more obvious when we colour the points by normalised Length (Fig. 5.10), where a smooth transition from the shortest sequence to the longest is revealed (subject to a discontinuity occurring when embedding high dimensional objects into a low dimensional space, see [138]

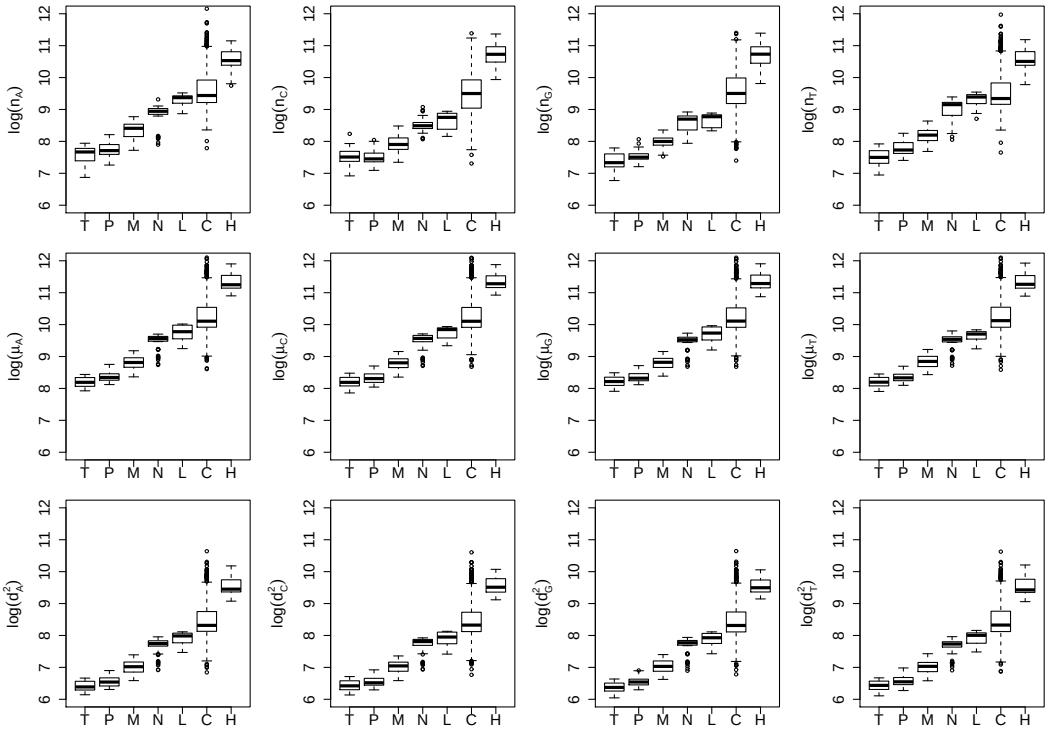


Figure 5.4: The same as Fig. 5.3 but for genome sequences with ICTV Order labels.

for technical details).

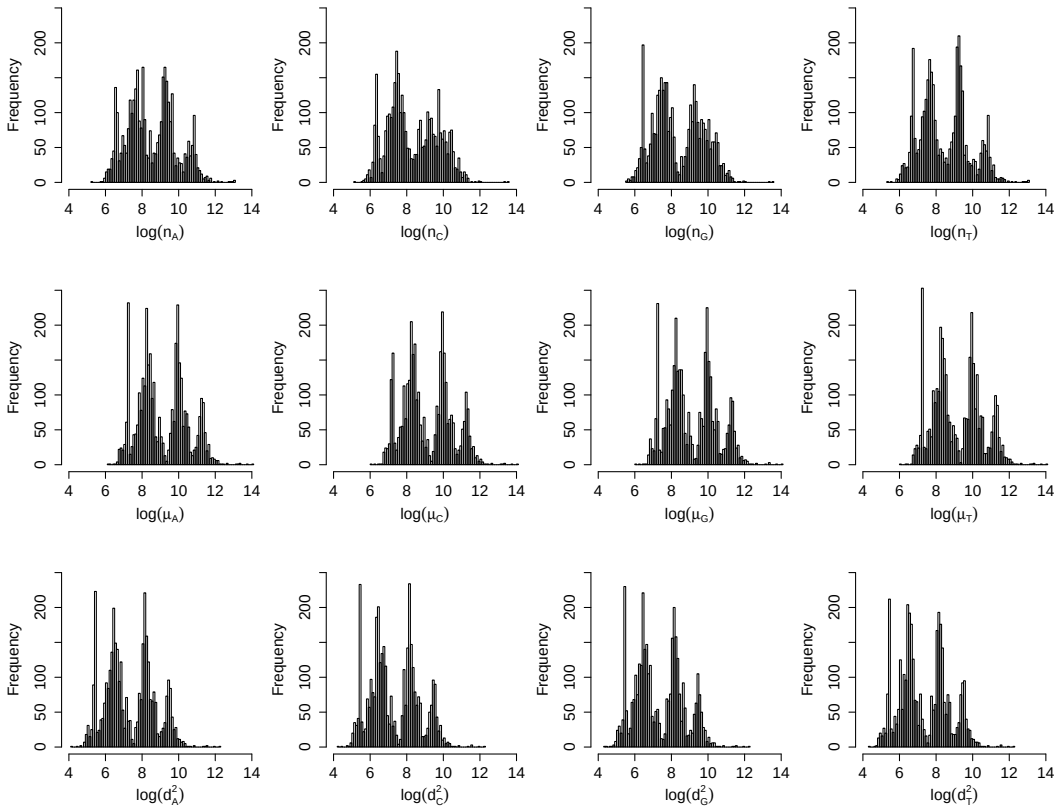


Figure 5.5: Histograms of genome sequence-related summary statistics for Baltimore Class-labelled viruses.

The statistics depicted are the natural log transformed value of count (n_A, n_C, n_G, n_T), mean position in the sequence $(\mu_A, \mu_C, \mu_G, \mu_T)$ and normalised variance of the position $(d_A^2, d_C^2, d_G^2, d_T^2)$ of each base.

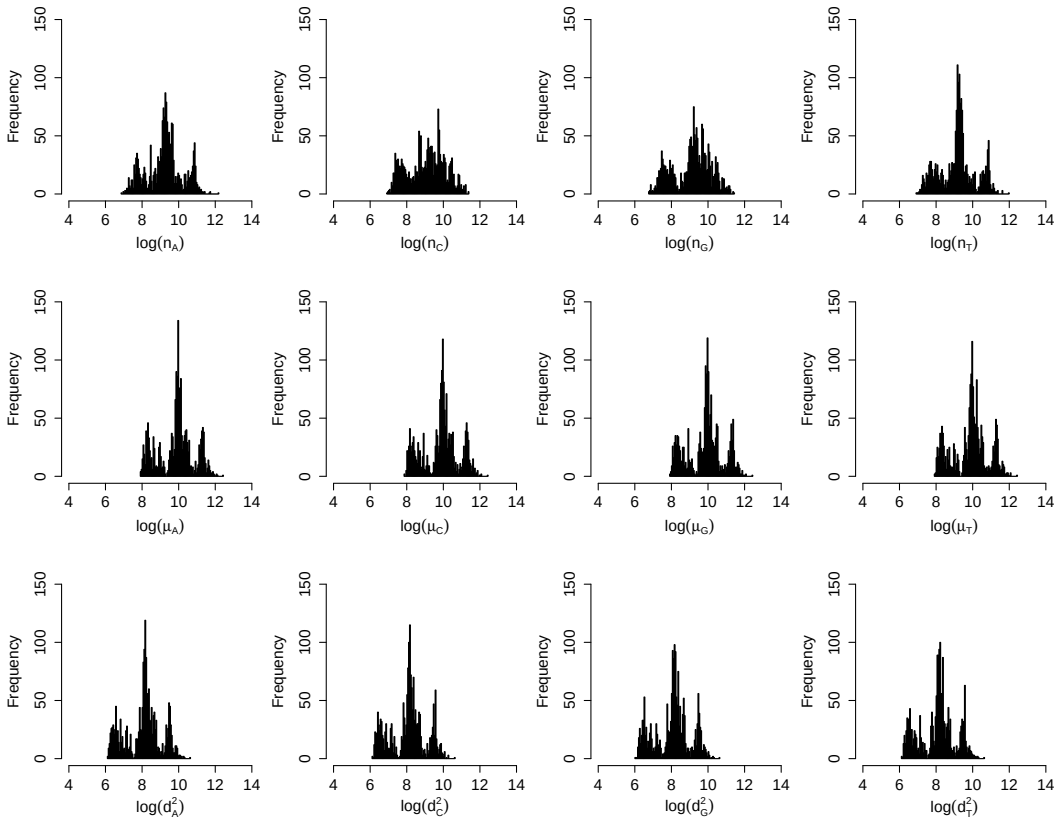


Figure 5.6: The same as Fig. 5.5 but for virus with ICTV Order labels.

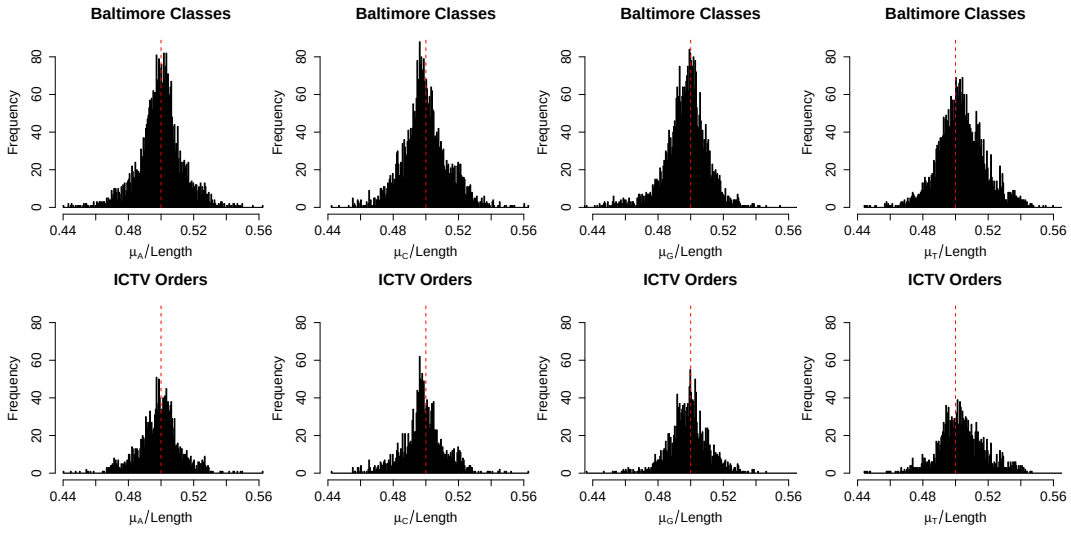


Figure 5.7: Histograms of mean nucleotide position normalised by genome length for viruses with Baltimore Class labels and ICTV Order labels.

The red dashed line indicates a normalised mean nucleotide position of 0.5.

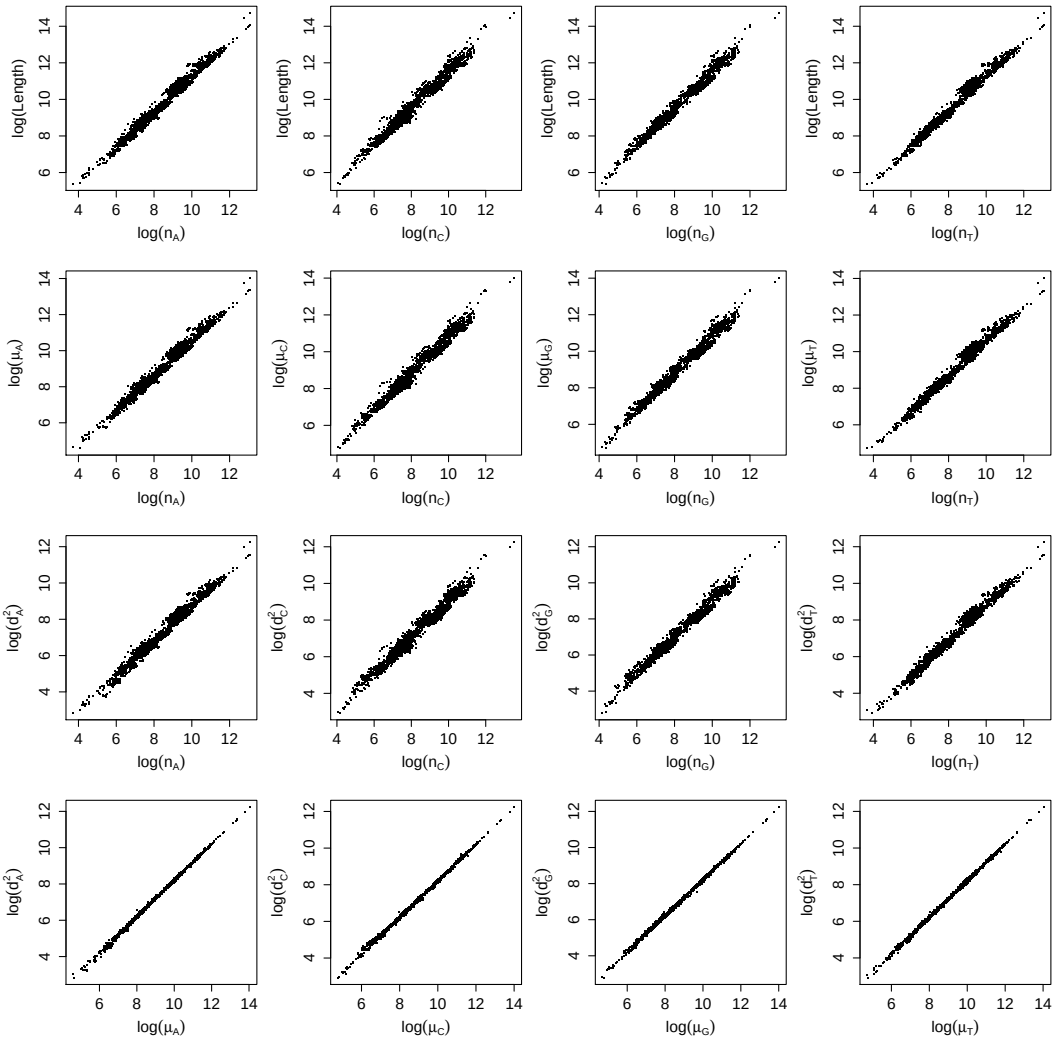


Figure 5.8: Scatter plots between summary statistics for genome sequences.

The Columns are for nucleotides A, C, G and T respectively; rows are for Length vs count, mean position vs count, normalised variance vs count, and mean position vs normalised variance. Values in the plots are the natural log of the original values.

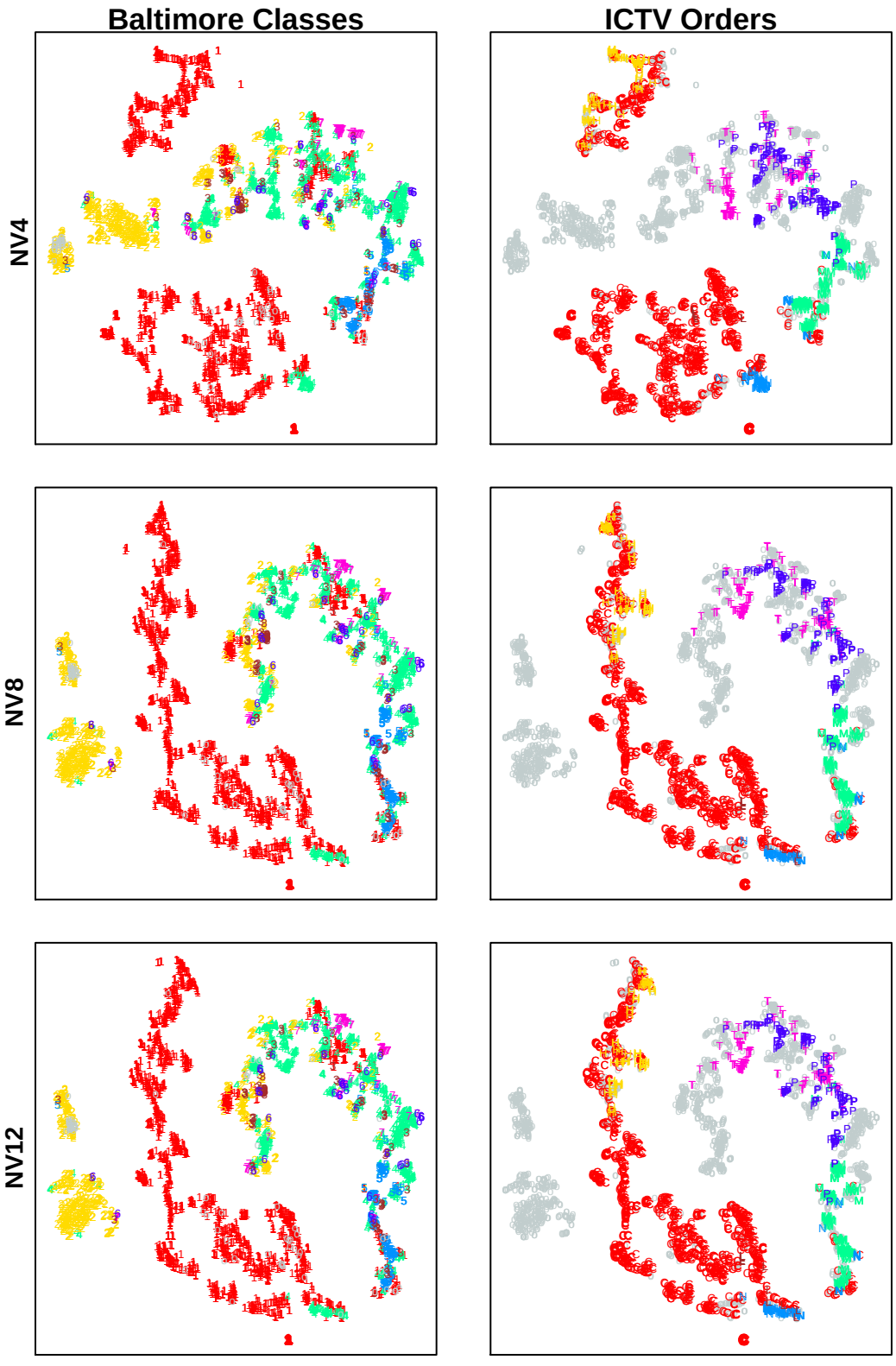


Figure 5.9: t-SNE visualisation of sequences using NV and its derivatives coloured by taxonomic classes.

Colours represent different classes: grey represents unlabelled viruses; red, yellow, brown, green, light blue, dark blue, pink are for Baltimore Classes I-VII respectively, and ICTV Order C, H, L, M, N, P, T respectively. Point styles for unlabelled viruses are circles, and for labelled viruses abbreviations of their class names (Arabic numbers 1-7 are used for the seven Baltimore Classes instead of Roman numerals I-VII).

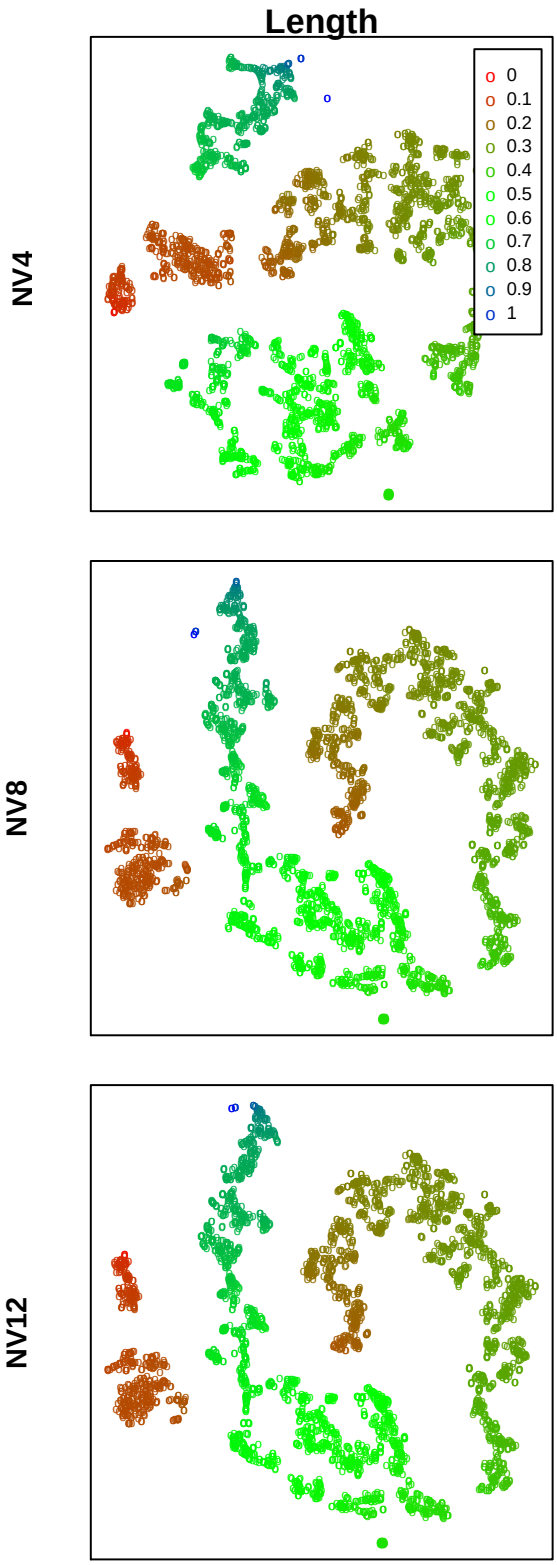


Figure 5.10: t-SNE visualisation of sequences using NV and its derivatives coloured by Length.

Colours represent normalised Length. Normalisation is done by scaling the natural log transformed Length to the range between 0 and 1, corresponding to the colours red and blue.

5.2.2 Features based on words

In this section, we visualise the properties of word and compression based features.

Features based on present words. When we extend NV to k-mer NV, a much longer feature vector can be produced with a significant amount of additional information. The resulting t-SNE plot gives cleaner class boundaries with less overlapping (Fig. 5.11).

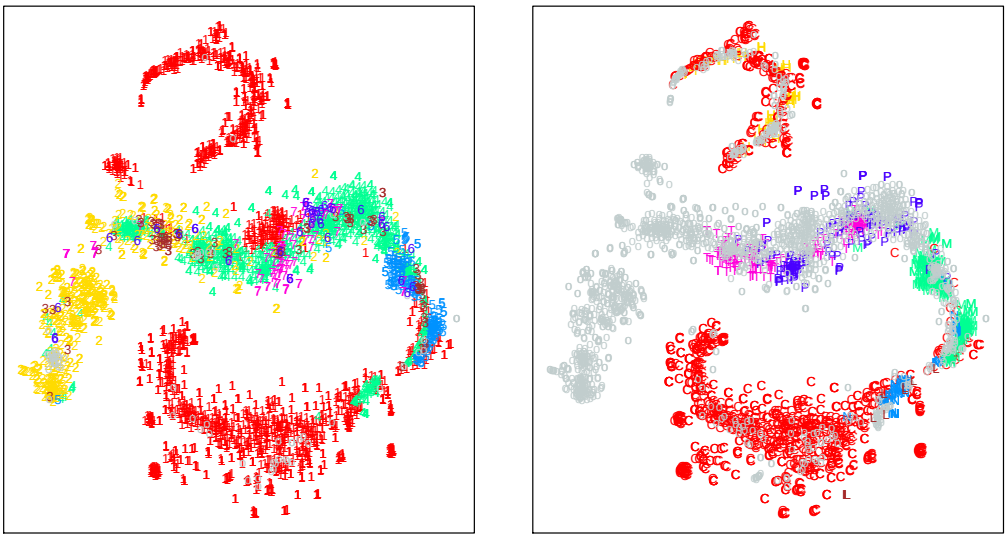


Figure 5.11: t-SNE visualisation of the sequences using 6-mer NV coloured by taxonomic classes.
Colours represent different classes: grey represents unlabelled viruses; red, yellow, brown, green, light blue, dark blue, pink are for Baltimore Classes I-VII (left) respectively, and ICTV Order C, H, L, M, N, P, T (right) respectively. Point styles for unlabelled viruses are circles, and for labelled viruses abbreviations of their class names (Arabic numbers 1-7 are used for the seven Baltimore Classes instead of Roman numerals I-VII).

In addition to the commonly used ACGT nucleotide alphabet, we also study the properties using other alternatives, the SW-RY-MK alphabet. The box plots in Fig. 5.12 show that the distributions of the ratios between the two types of nucleotides for all three alphabets are distinguishable for different Baltimore Classes or ICTV Orders, though the difference is smaller compared to the ACGT alphabet. The median of the ratios are around one, with that of S/W slightly below and that of R/Y slightly above.

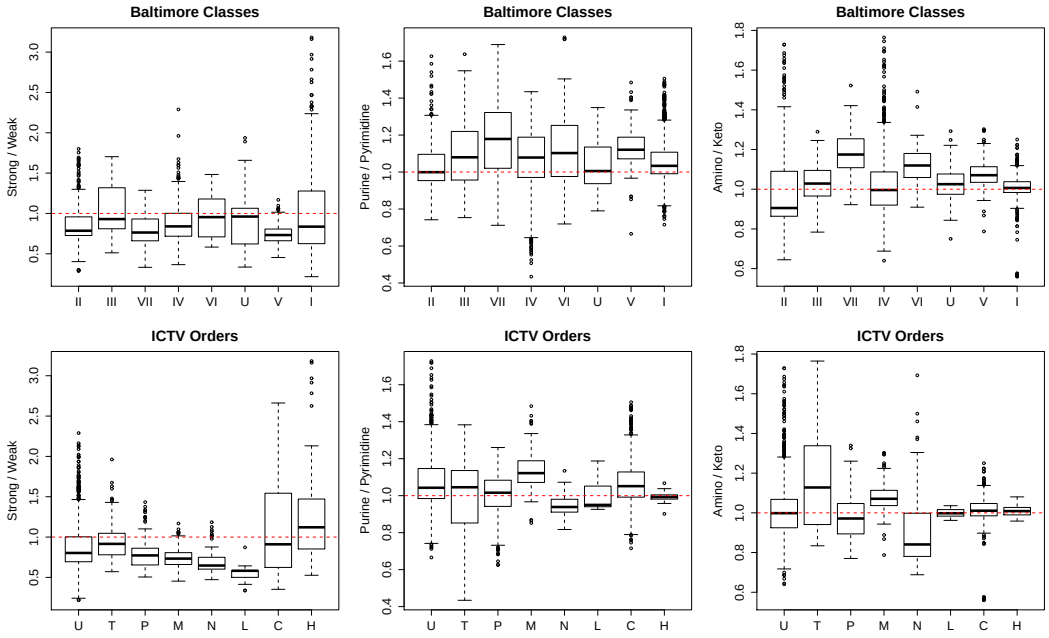


Figure 5.12: Box plots of the nucleotide content of virus genome sequences. Box plots for the ratio of strong (G+C) to weak (A+T), purine (A+G) to pyrimidine (C+T), and amino (A+C) to keto (G+T) nucleotides in a genome. Each panel shows eight box plots ordered by increasing median Length: seven for known taxa (Baltimore Class or ICTV Order) and one for unlabelled viruses ("U"). The horizontal red dashed line indicates a ratio of 1.0.

Fig. 5.13 reveals various patterns of nucleotide composition. We can observe that a clear stratification of viruses from different classes is produced by Length. The ratios of two types of nucleotides of different viruses are mostly concentrated at one but disperse on both sides. The degree of dispersion differs between classes and alphabets. For example, the pattern of S/W of Baltimore Class I and ICTV Orders H and C, M/K of Baltimore Classes IV and II and ICTV Orders N and T significantly extend towards a large ratio, whereas the pattern of M/K of Baltimore Class I and ICTV Order C are roughly symmetric about one. R/Y is the most symmetric and S/W is the least. Our finding are consistent with previous works that focus on specific viruses categories [142] and codon usage [143, 144].

The relationship between ratios of the three alphabets can be displayed using ternary coordinates (Fig. 5.14 - 5.16). A point in the plot corresponds to a virus and its precise location depends on the physicochemical property used to group the

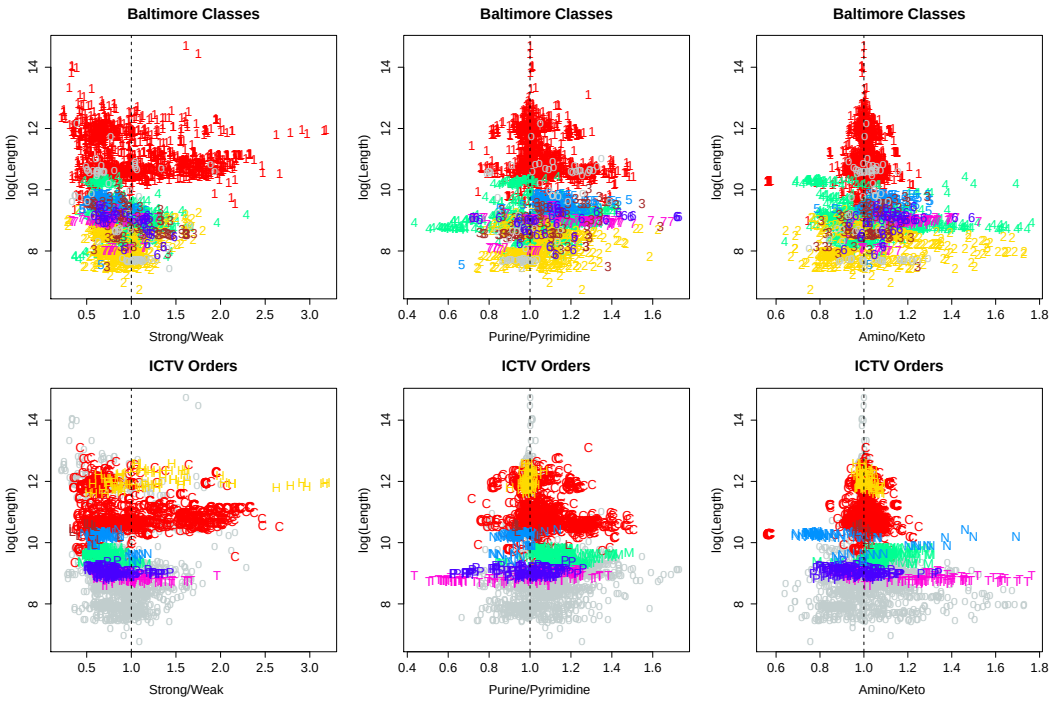


Figure 5.13: Plots of overall nucleotide composition against genome length. The natural log transformed value of Length plotted against the ratio of strong (G+C) to weak (A+T), purine (A+G) to pyrimidine (C+T), and amino (A+C) to keto (G+T) nucleotides in a genome. Points corresponding to labelled virus genomes are coloured and marked by taxonomic scheme (Baltimore Class or ICTV Order) whereas unlabelled ones are depicted as open grey circles.

four bases: strong/S (G, C) or weak/W (A, T), purine/R (A, G) or pyrimidine/Y (C, T), and amino/M (A, C) or keto/K (G, T). Vertices labelled S, R, and M indicate the percent of nucleotides in a genome characterised as Strong, puRine and aMino respectively.

We can observe that the values are mainly concentrated at the median line connecting vertex S to its opposite edge, which indicates that S has the largest variance but R is mostly similar to M. Different classes tend to favour different ratios and display different patterns. For example, the region M $>$ S $>$ R is dominated by Baltimore Class IV (Fig. 5.15) and ICTV Order T (Fig. 5.16). ICTV Order H is concentrated at the median line and spans a large range, whereas M and N favour smaller S.

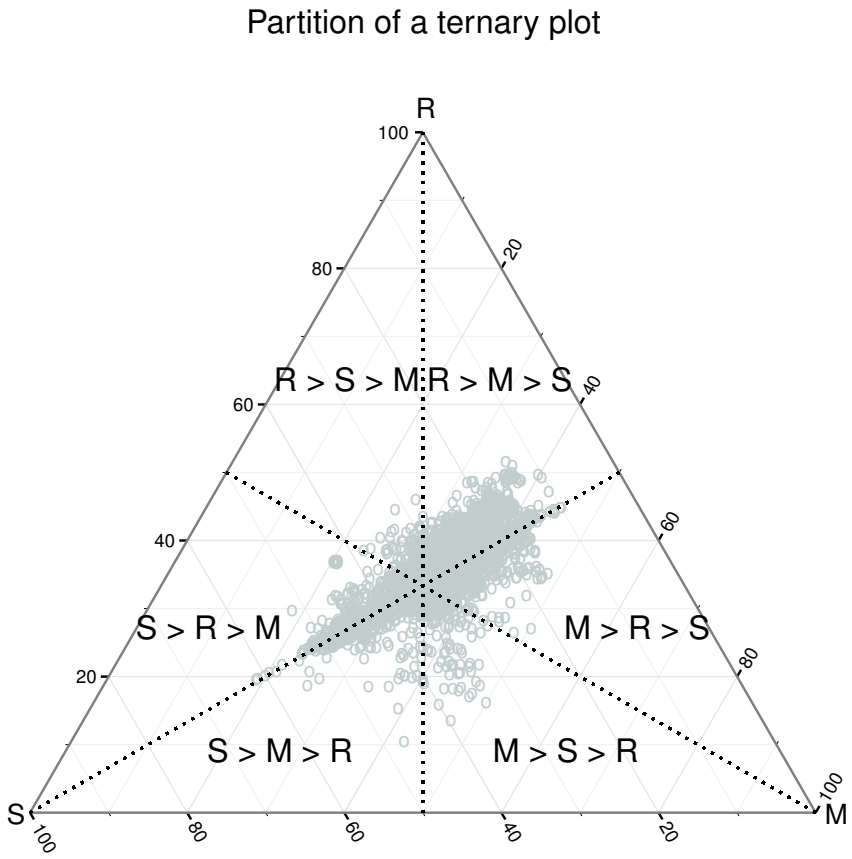


Figure 5.14: Ternary plot of the nucleotide composition using three 2-letter alphabets (partition of the plot).

The plot is partitioned into six regions and the type of composition bias is exhibited by genomes located in each region; for instance, of the virus genomes analysed here (grey circles), all those in the S > R > M region have the same composition pattern %(G+C) > %(A+G) > %(A+C).

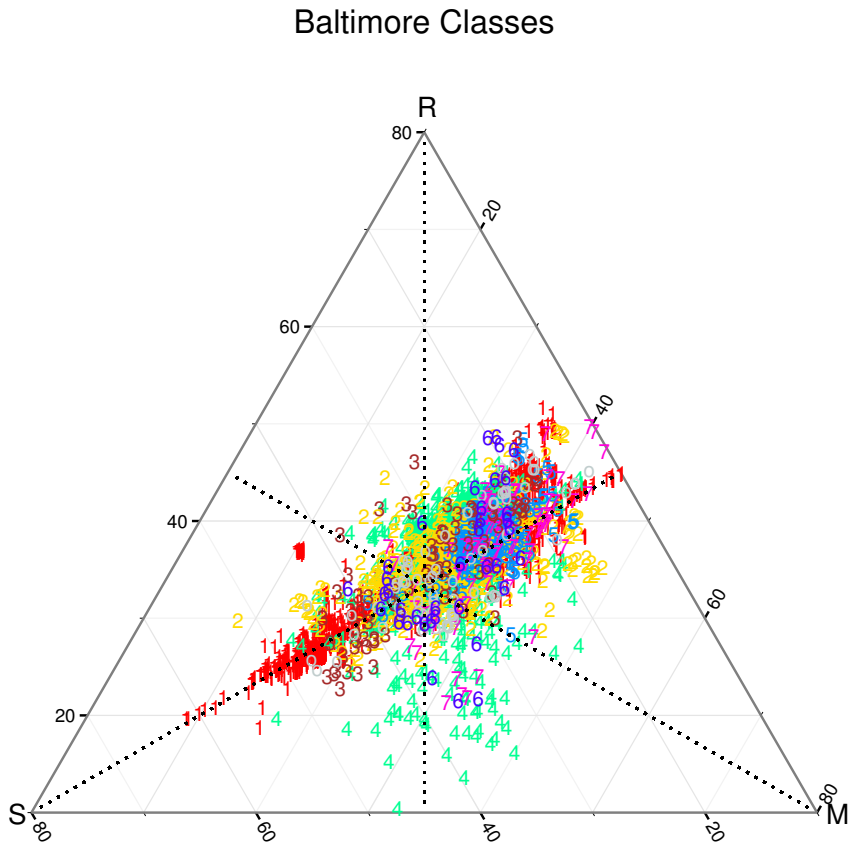


Figure 5.15: Ternary plot of the nucleotide composition using three 2-letter alphabets (Baltimore Classes).
Points corresponding to labelled virus genomes are coloured and marked according to Baltimore scheme whereas unlabelled ones are depicted as open grey circles.

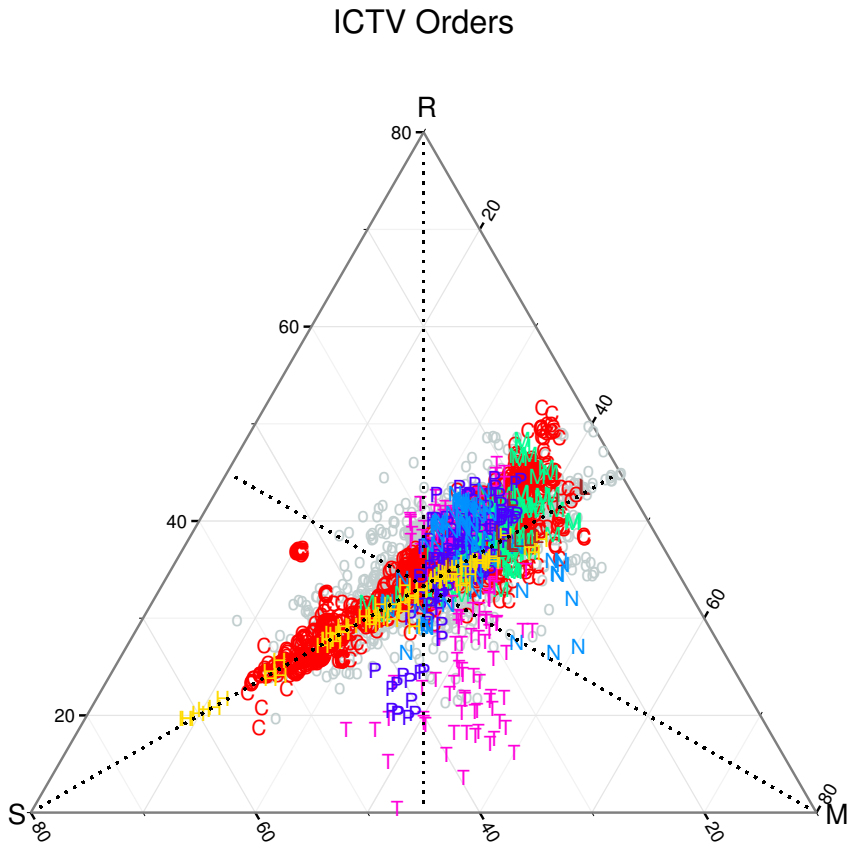


Figure 5.16: Ternary plot of the nucleotide composition using three 2-letter alphabets (ICTV Orders).

Points corresponding to labelled virus genomes are coloured and marked according to ICTV Order scheme whereas unlabelled ones are depicted as open grey circles.

Features based on absent words. Fig. 5.17 shows the trend of the percentage of MAW as the length of MAW increases. We can observe that for both Baltimore Class and ICTV Order labelled sequences, the number of MAW is mostly zero for short words, quickly increases when word length reaches about five, then peaks at around nine, and finally quickly decreases and remains near zero. This trend is consistent with the finding in [118] and is believed to be universal for species from different kingdoms. This is a favourable property of MAW since its number does not grow exponentially with word length, as opposed to that of absent words, making it easier to analyse.

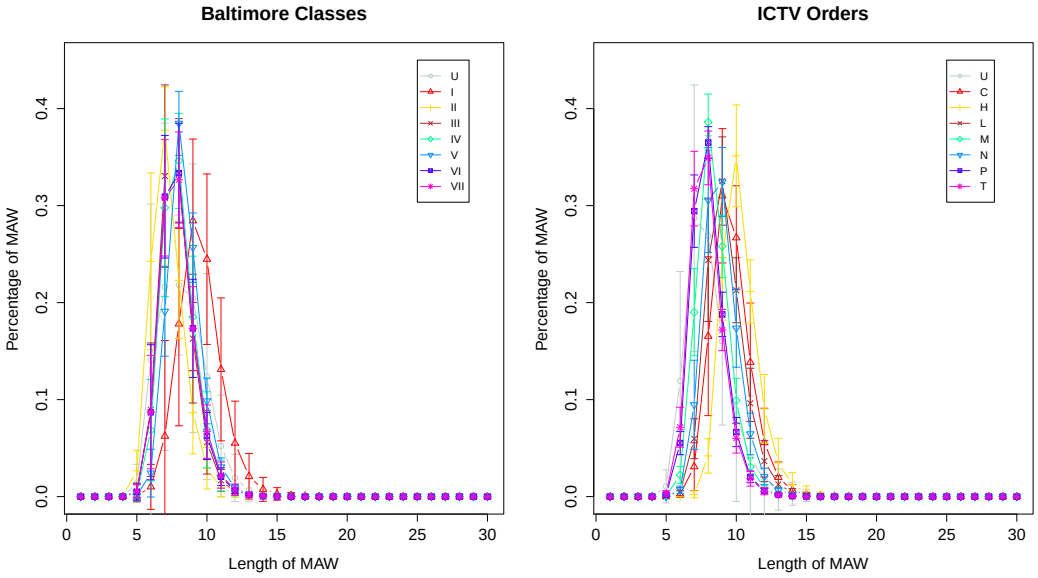


Figure 5.17: Distribution of MAW in genome sequences.
Error bar plots of the number of MAW vs length of MAW in each Baltimore Class (left) and ICTV Order (right).

Fig. 5.18 shows the relationship between genome length and MAW. A strong linear correlation exists between genome length and the number of MAW (left column). The length of the shortest MAW is indistinguishable for viruses from different classes (middle column), but that of the longest MAW varies significantly (right column). This observation is also consistent with the finding in [118] and is the reason that the longest MAWs are of particular interest when studying different organisms.

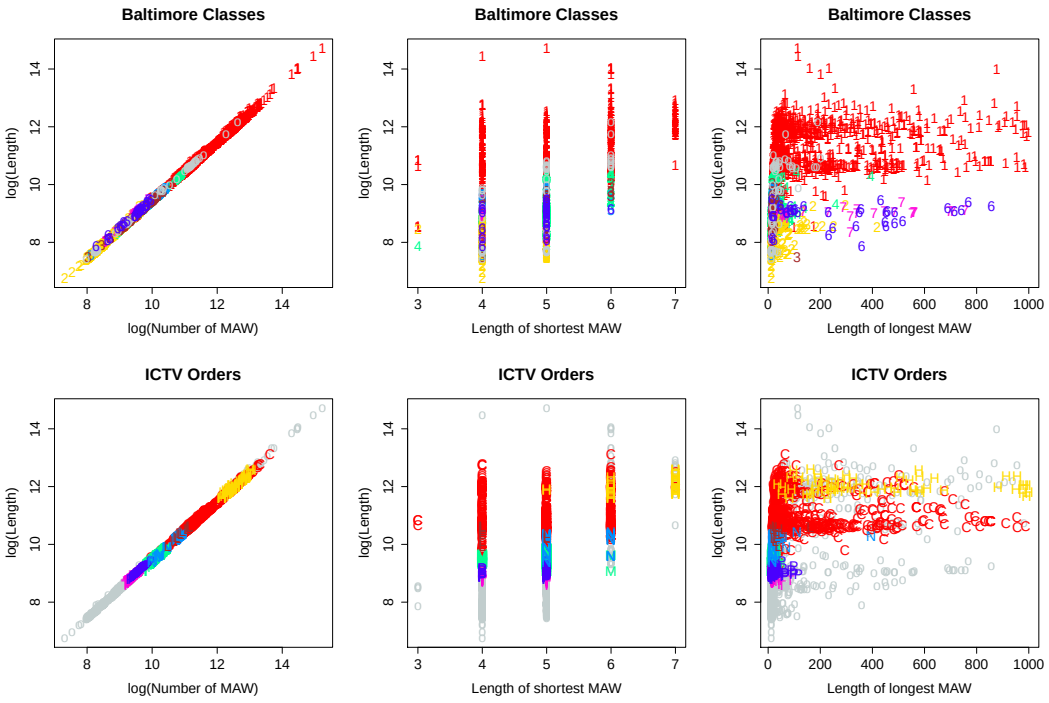


Figure 5.18: The relationship between genome length and MAW.
Scatter plot of natural log transformed Length vs log transformed number of MAW (left column), log transformed Length vs length of the shortest MAW (middle column), and log transformed Length vs length of the longest MAW (right column) for Baltimore Classes (top row) and ICTV Orders (bottom row). Point styles indicate class names as in Fig. 5.11.

5.2.3 Features based on compression

Fig. 5.19 and 5.20 show that the summary statistics of compression ratios for different Baltimore Classes or ICTV Orders appear differently using different general-purpose or DNA specific tools, but are less distinguishable than that of nucleotide statistics (Length, NV4, NV8 and NV12) (Fig. 5.2 - 5.4). The distributions for different classes using DNA specific tools appear more distinguishable than those obtained using general-purpose tools. The lowest average compression ratio for all sequences in the dataset from a general-purpose tool is achieved by bzip2 (0.290), which, however, is only slightly better than the others (gzip: 0.315, xz: 0.303, zip: 0.359). In addition, DNA specific compression tools reduce compression ratio moderately compared to general purpose ones except for LEON, which gives the best ratio among all methods in the study with a lowest average compression ratio of 0.142 (the other two are DELIMINATE: 0.283, and MFCompress: 0.273).

Histograms (Fig. 5.21 - 5.22) show that the distributions of the compression ratios tend to skew towards the left side and the degree of skewness is more prominent for DNA specific tools. The histograms tend to contain one major peak and two relatively minor ones.

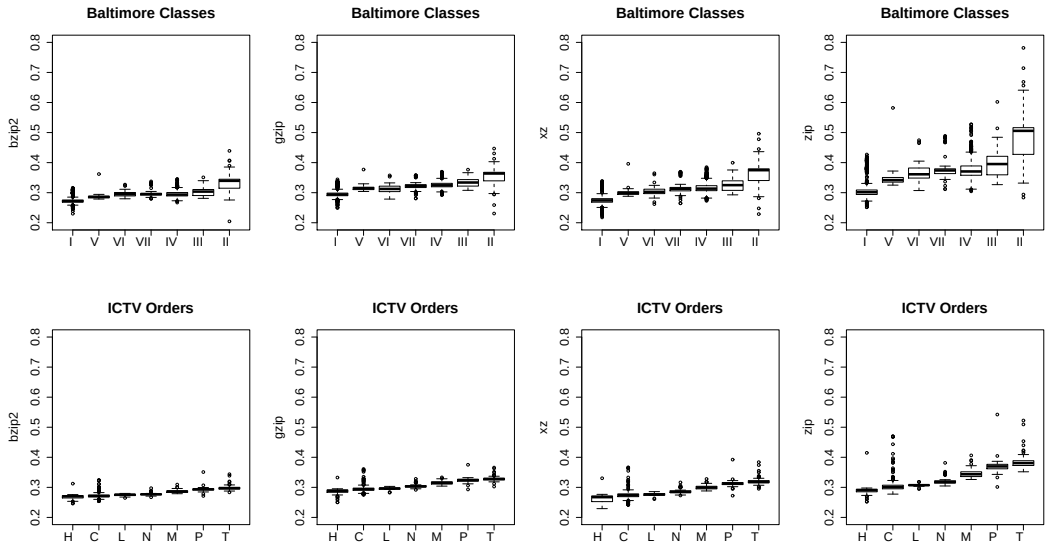


Figure 5.19: Box plots of compression ratio achieved by general-purpose compression tools.

Baltimore Classes and ICTV Orders are ordered by increasing median compression ratio.

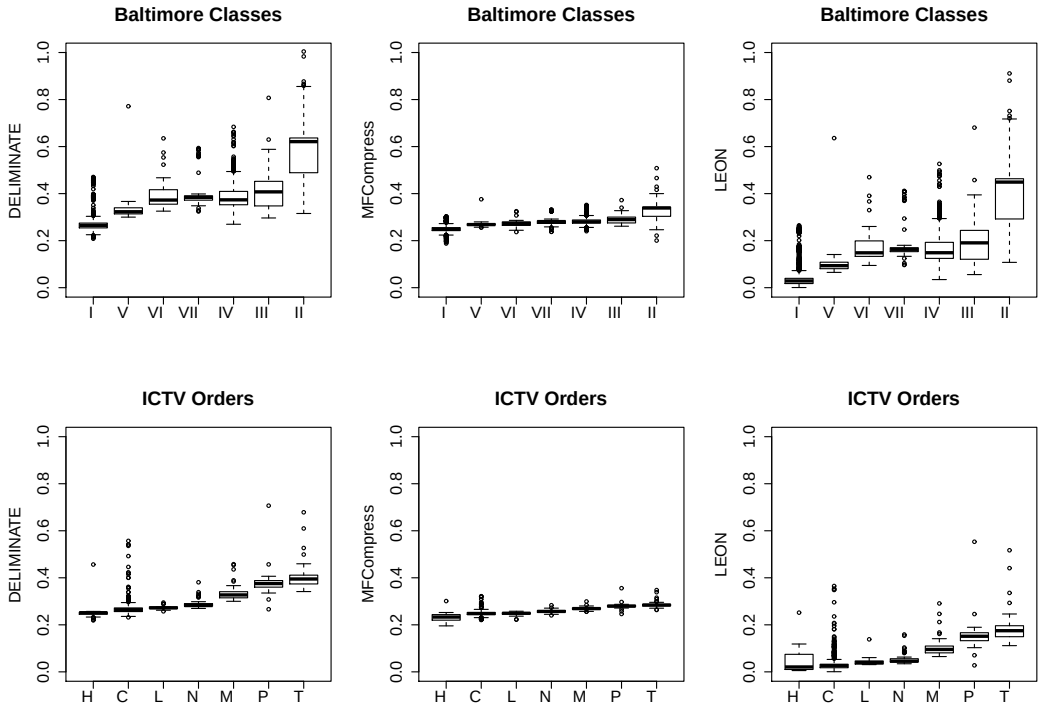


Figure 5.20: Box plots of compression ratio achieved by DNA-specific compression tools.

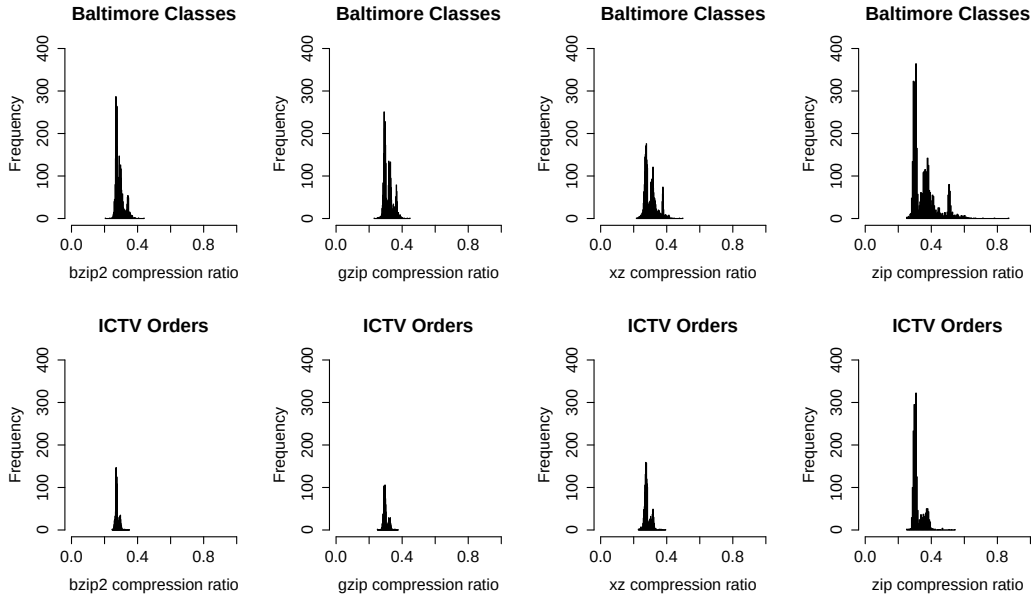


Figure 5.21: Histogram of compression ratio achieved by general-purpose compression tools.

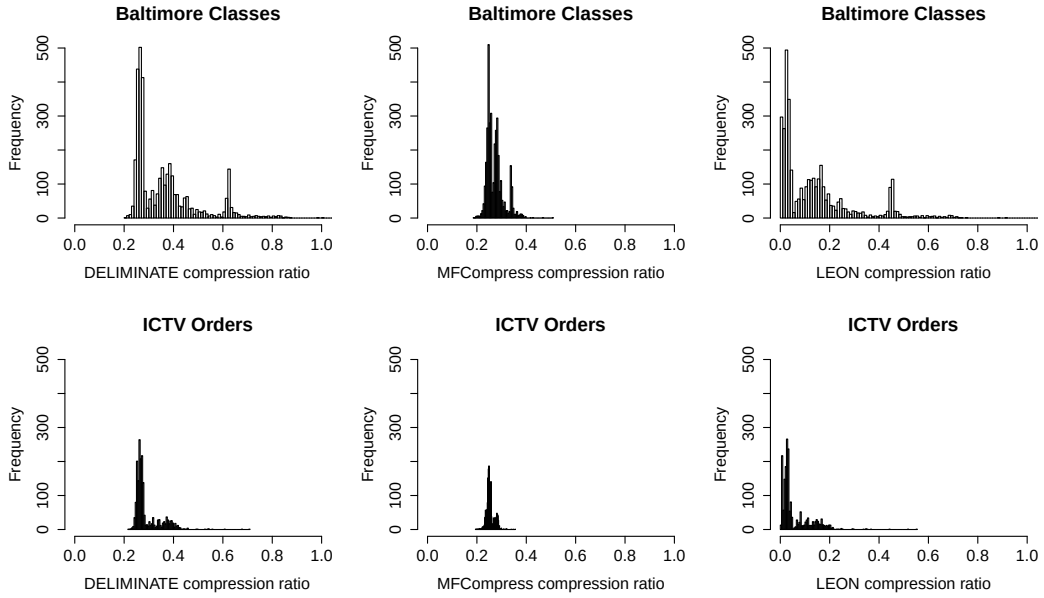


Figure 5.22: Histogram of compression ratio achieved by DNA-specific compression tools.

5.3 Summary

In this chapter, we studied the properties of our virus genome sequence dataset using various visualisation techniques. We have shown that the distributions of global composition- and location-related statistics of nucleotides or words, as well as MAW and compression ratios of the sequences in the dataset appear distinguishable between different classes, ICTV Orders in particular, and Baltimore Classes to a lesser extent.

Length of a virus genome sequence is sometimes used as one of the taxonomic criteria in the ICTV scheme, but not explicitly in the Baltimore scheme. The predictive power and importance of Length in differentiating classes in both schemes is an interesting finding.

The distributions of nucleotide counts, mean position and normalised variance from all sequences in the dataset are multi-modal, but become uni-modal when normalised by Length. These statistics have strong linear correlations with Length as well as with each other.

The distributions of the lengths of MAW do not grow exponentially as Length does, but instead start from near zero, quickly reach a peak of between 5 and 10, and then rapidly decrease to near zero again. The length of the longest MAW of sequences vary significantly between classes, and hence can be used to distinguish them.

The compression ratio of the sequences reflects the complexity of their underlying patterns and can be used to distinguish different classes. DNA-specific compression tools give better compression performance than general-purpose ones and their compression ratios can be better features to distinguishable sequences from different classes.

In addition, we used t-SNE, an unsupervised dimensionality reduction technique, to visualise all sequences as two-dimensional plots. Results show that the technique is able to cluster viruses from the same class and separate those from different classes. The results also confirm the important role Length plays in distinguishing sequences.

Chapter 6

Multi-class classification using nucleotide-based features

In this chapter, we study the predictive power of global statistics of nucleotides in virus taxonomy. The aim is to extend previous work [52] by investigating the contributions from the components of NV and exploiting their predictive powers when combined with more sophisticated classifiers.

We first assess the performance in predicting the Baltimore Classes and ICTV Orders using different combinations of features and classifiers, and examine the contribution of different components of the NV12 to classification performance. Then, we identify sequences that tend to be misclassified by different classifiers and propose an approach to reduce misclassification of these viruses. Next, we predict viruses in the dataset that are currently unlabelled by Baltimore Classes or ICTV Orders using the best combination of experimental factors. Finally, we explore the performance of NV12 in the task of predicting virus hosts.

6.1 Methods

6.1.1 Preprocessing

To construct features, we ignore ambiguous nucleotides in a virus genome sequence when computing global composition-based statistics (nucleotide count n_s) but retain them when calculating global location-based statistics (mean position and variance of a nucleotide). We use these summary statistics to create numerical descriptors of

a sequence (Table 4.1).

To improve statistical stability and performance, the features in the SVM are normalised so that each variable in the feature vector has zero mean and unit variance. For the k-NN and RF classifiers, the original features are used as similar normalisation does not improve performance.

6.1.2 Parameter optimisation

To optimise parameters for each classifier, we use 5-fold Cross-Validation (CV) to tune parameters over a range of values (Table 6.1) in a grid search fashion using viruses in the training set. The combination of values giving the lowest mean error rate are then selected. The software packages used in the study are shown in Table 3.1.

Classifier	**Range of parameter values**
k-NN	$k \in \{1,2,3,...,10\}$
RF	$ntree \in \{1,201,401,...,1001\}$ $mtry \in \{1\}$
Radial-SVM	$C \in \{2^5,2^6,2^7,...,2^{17}\}$ $\gamma \in \{2^{-18},2^{-16},2^{-14},...,2^{13}\}$

Table 6.1: Range of parameter values used during cross-validation.

6.2 Predicting the Baltimore Class and ICTV Order of non-segmented viruses

6.2.1 Design

In this section, we aim to study the performance of predicting the Baltimore Class and ICTV Order of a virus genome sequence using different combinations of features and classifiers. We extend earlier work [52] by applying two other machine learning techniques, Support Vector Machines (SVM) and Random Forests (RF). We also provide a comprehensive analysis of the properties of the dataset, examine the contribution of different components of the NV12 embedding to classification performance, and explain their behaviour during classification.

The dataset used in this section consist of two parts. One for Baltimore

Class prediction, which contains the 3,625 non-segmented Baltimore Class labelled viruses, and the other for ICTV Order prediction, which contains the 1,865 non-segmented ones (Table 2.1).

Two sets of parallel experiments are conducted, one for Baltimore Class prediction and the other for ICTV Order prediction. For the viruses used in each set of experiments, we randomly divide them into training and testing sets in the ratio 75% : 25%. For each combination of feature and classifier, we first train the classifier and optimise its parameters using features of viruses in the training set, and then test using the testing set. This procedure is repeated 10 times with random seeds, each giving a different training/testing set division. Finally, we report the mean and standard deviation of the testing error rates over the 10 repeats to summarise the classification performance on the overall dataset, and use the confusion matrix to detail the behaviour of individual classes.

6.2.2 Overall classification performance

Table 6.2 and Fig. 6.1 show the overall classification error rates using the four classifiers and four features in predicting the Baltimore Class and ICTV Order of a virus genome sequence. Our observations are as follows. First, errors for ICTV Orders are lower than those for Baltimore Classes for all combinations of features and classifiers. Second, errors decrease from Length to NV12 for all classifiers and in both schemes. Classification based on Length already gives respectable performance, and NV4 gives a significant improvement, though more complex NV only give a slight improvement over NV4. Third, the best classifiers for features listed in the table are either k-NN or SVM. k-NN is commonly used for similar studies, and is simple but performs well. SVM is more sophisticated compared to k-NN and gives the lowest error rate in the table. Fourth, NV12 with SVM is the best combination for both Baltimore Class and ICTV Order experiments in terms of mean error rates. Fifth, parameter optimisation gives slight improvements to k-NN and RF, but significant improvements to SVM. Indeed, SVM gives the highest error rates without preprocessing or parameter optimisation, whereas k-NN and RF perform well even without these.

Baltimore Classes				
Feature vector	**Classifier**			
	1-NN	k-NN	SVM	RF
Length	0.234 ± 0.010	**0.204 ± 0.009**	0.205 ± 0.010	0.267 ± 0.049
NV4	0.089 ± 0.011	0.089 ± 0.011	**0.084 ± 0.009**	0.090 ± 0.007
NV8	0.076 ± 0.010	0.076 ± 0.010	**0.072 ± 0.009**	0.078 ± 0.006
NV12	0.075 ± 0.010	0.075 ± 0.010	**0.065 ± 0.009**	0.079 ± 0.008

ICTV Orders				
Feature vector	**Classifier**			
	1-NN	k-NN	SVM	RF
Length	0.171 ± 0.019	**0.137 ± 0.013**	0.144 ± 0.012	0.160 ± 0.021
NV4	**0.059 ± 0.010**	**0.059 ± 0.010**	0.073 ± 0.009	0.062 ± 0.010
NV8	**0.051 ± 0.011**	**0.051 ± 0.011**	0.053 ± 0.009	0.055 ± 0.008
NV12	0.049 ± 0.007	0.049 ± 0.007	**0.038 ± 0.008**	0.053 ± 0.008

Table 6.2: Classification error rates of different combinations of features and classifiers.

The mean and standard deviation of classification error rates using features Length, NV4, NV8 and NV12, and classifiers 1-NN, k-NN, SVM and RF. Results from 1-NN are also shown for comparison with those from similar papers [52]. The lowest mean error rates for each feature vector are highlighted in bold.

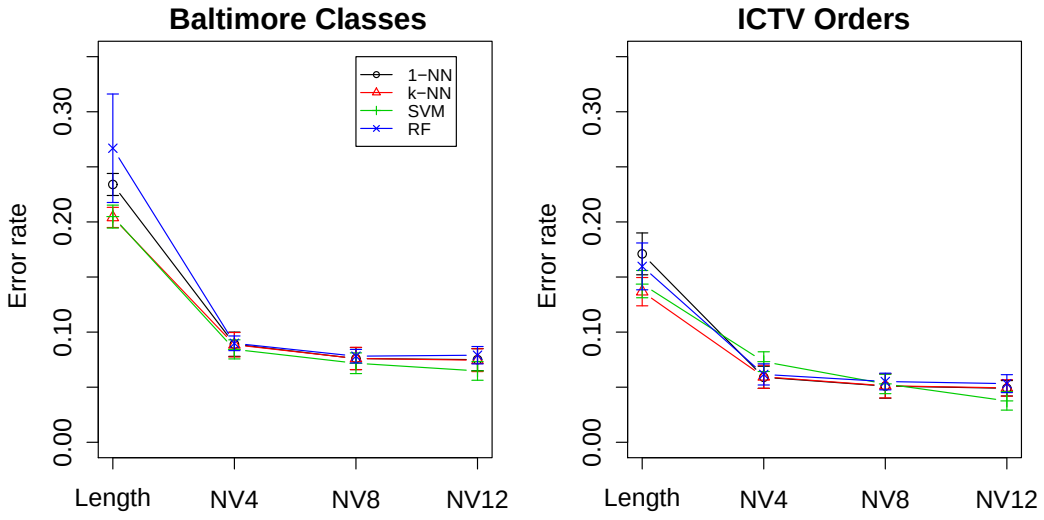


Figure 6.1: Error bar plots of classification errors shown in Table 6.2.

6.2.3 ICTV Orders are easier to predict than Baltimore Classes

Results from Table 6.2 confirm our previous notion based on Fig. 5.2 - 5.4 that ICTV Orders are easier to distinguish than Baltimore Classes. Membership between

the two taxonomy systems shown in Table 2.2 also support this notion as each ICTV Order is a "finer" subset of a Baltimore Class. The ICTV scheme groups viruses based on a variety of criteria including the similarity of genome sequences, whereas the Baltimore scheme classifies them based only on genome type and method of replication. Therefore, the Baltimore scheme is coarser and tends to have larger intra-class variation and larger inter-class similarity in its genome sequences, and hence different classes are less separable using genome sequence based methods.

6.2.4 Simple features can give respectable classification performance

Table 6.2 suggests that Length can already give respectable classification performance, NV4 performs nearly as well as the full NV12, and more complex features only give a marginal improvement.

The importance of Length is supported by Fig. 5.2, where the distributions for different classes appear separable. The good performance of NV4, NV8 and NV12 is also indicated by the box plots of counts, mean position and normalised variance (Fig. 5.3 - 5.4). The finding that improvements from NV4 with NV8 or NV12 are small is explained by the correlation plots between Length, counts, mean position and normalised variance (Fig. 5.8), since all pairs exhibit strong linear correlation, with that between the mean position and normalised variance being the strongest.

The representational ability of NV related simple features can also be demonstrated using t-SNE plots (Fig. 5.9), where distinct clusters are revealed in an unsupervised setting. Length plays an important role in the embedding procedure. Classes having similar Length distributions in Fig. 5.2 are neighbours in the t-SNE plots, with a smooth transition from the shortest sequence to the longest (subject to a discontinuity occurring when embedding high dimensional objects into low dimensional space, see [138] for technical details). The high degree of similarity between the embeddings produced from NV8 and NV12 indicates that the normalised variance of nucleotide positions add a negligible amount of information over the mean positions, which can be explained by the strong correlation revealed in Fig. 5.8.

The observations from Chapter 5 support the results shown in Table 6.2 that

distinct genome compositions and nucleotide arrangements of different viruses can be summarised by simple features. Due to the strong correlation between feature variables, the 1D feature Length already conveys a large amount of information about the genome sequences. NV4, the decomposition of Length into finer detail, contains a considerable proportion of the amount of information a full NV12 has, though more complex features add marginal information.

6.2.5 Small classes tend to be confounded with large ones with similar Length distribution

As a further step to analyse the causes of the overall classification errors shown in Table 6.2, we examine the performance of individual classes by plotting the confusion matrix of the classification results (Fig. 6.2, 6.3). For both Baltimore Class and ICTV Order experiments, there is a clear correlation between the size and error rate of a class: larger classes generally give higher accuracy than smaller ones. For example, the accuracy of the larger Baltimore Classes I, IV, II, V are consistently above 0.6 when using Length with k-NN or SVM, whereas those of smaller classes VII, III, VI are generally below 0.2; a similar contrast exists between the larger Orders C, M, T, P and smaller ones N, H, L.

However, errors of certain classes can differ significantly with others of similar size. For example, Baltimore Class III has a similar size as VII but its error rates are almost double that of VII. This may be caused by the Length distribution (Fig. 5.2) since that of III is very similar to the larger class IV and hence viruses from III have a high chance of being misclassified as IV (Fig. 6.2). Similarly, Orders H and N have matching size but H has a much higher chance of being misclassified as C.

The above analysis suggests that classification performance is mainly affected by two aspects: class size and Length distribution. Errors mainly arise from small classes, which tend to be confounded by larger ones with similar Length distribution. However, on the other hand, viruses from large classes are unlikely to be misclassified into small classes with similar Length distribution, since all the confusion matrices are non-symmetric with the bottom right having higher values than the upper left. An example is the viruses from small Baltimore Classes VII, III and

VI, which are frequently misclassified into the larger Class IV, but viruses from IV have a low chance of being misclassified.

In summary, larger classes have higher accuracy and higher false positive rates, whereas smaller ones have lower accuracy and higher false negative rates. These results are also reflected in the proximity between different classes shown in Fig. 5.9. Large classes such as the Baltimore Class I and ICTV Order C are well separated from the other classes. However, the Baltimore Classes III and VI overlap significantly with IV, and ICTV Order H and L are largely confounded by C, indicating higher errors in the confusion matrices.

6.2.6 Performance improves from NV12 more for small classes than large ones

Comparing the confusion matrices of different feature vectors from Length to NV12, we observe that the values along the diagonals increase (an increase in correct classification) and values in off-diagonal regions decrease (a decrease in misclassification) (Fig. 6.2, 6.3). In keeping with the overall classification errors, the improvement is noticeable going from Length to NV4, but modest onwards. However, the behaviour of small classes (e.g. the Baltimore Classes VII, III, VI and the ICTV Orders N, H, L) and large ones (e.g. the Baltimore Classes I, IV, II, V and the ICTV Orders C, M, T, P) differ. For small classes, the change is dramatic from Length to NV4, and still noticeable from NV4 to NV12, especially when using the SVM; however, for large ones, the improvement from Length to NV4 exists but is much less noticeable than for small classes, and almost negligible from NV4 to NV12. This suggests that a more complex feature improves the performance of small classes in particular, but large ones only marginally so. Owing to the bias in size, the obtained extra performance from small classes seems negligible in terms of the overall classification errors (see Table 6.2).

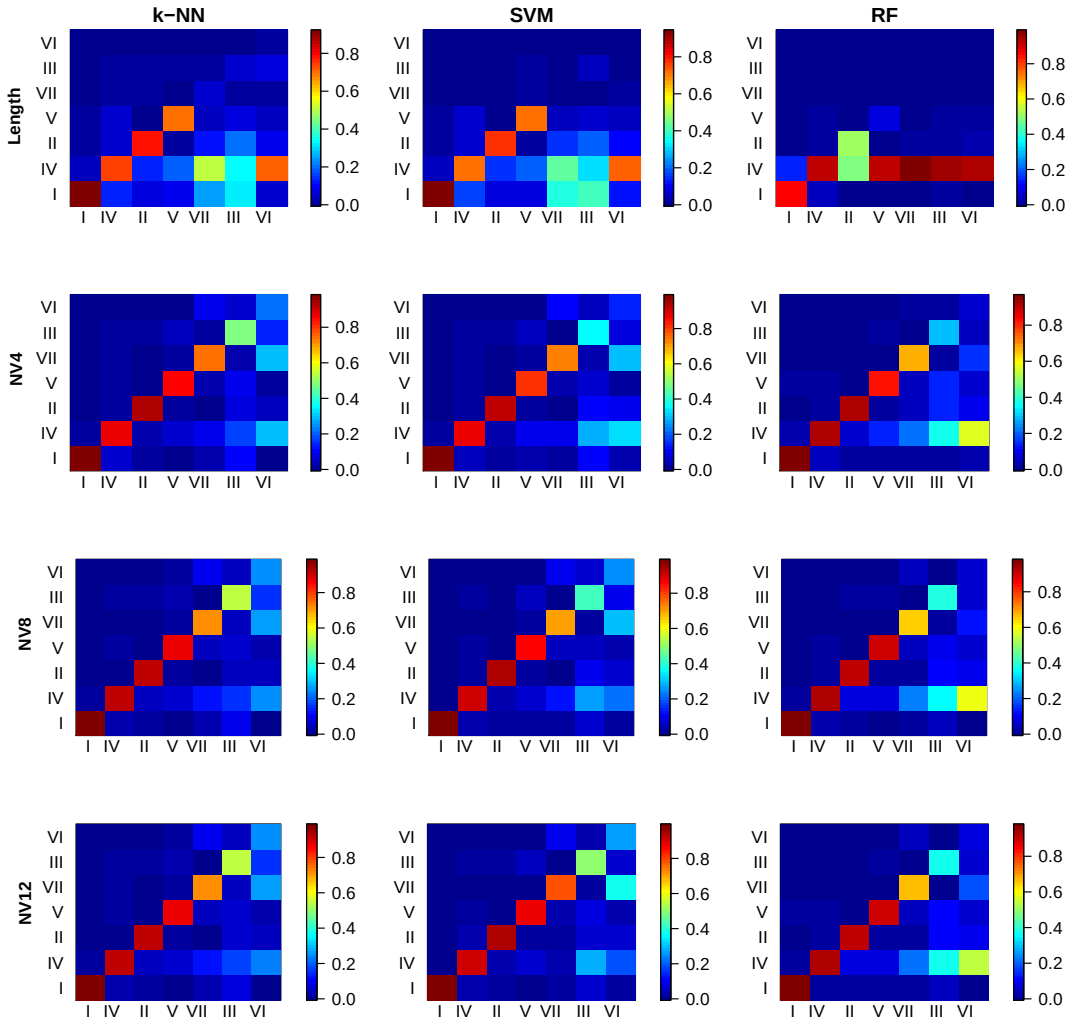


Figure 6.2: Confusion matrix for the Baltimore Class experiment.
In each subplot, the x-axis represents the true class labels and y-axis represents the prediction. Colours represent the percentage of viruses from class x being predicted as a member of class y. Diagonal entries represent rates of correct classification, where x equals y; off-diagonal entries are rates of misclassification. Classes are ordered in decreasing size from left to right and bottom to top.

6.3 Reducing misclassification of difficult viruses

6.3.1 Design

The aim of this section is two-fold. First, to identify "difficult viruses", which we define to be the viruses that are consistently misclassified by different classifiers. Their identification can suggest potential mislabelling in the current dataset or limitations of the feature representation. Improving performance on these viruses

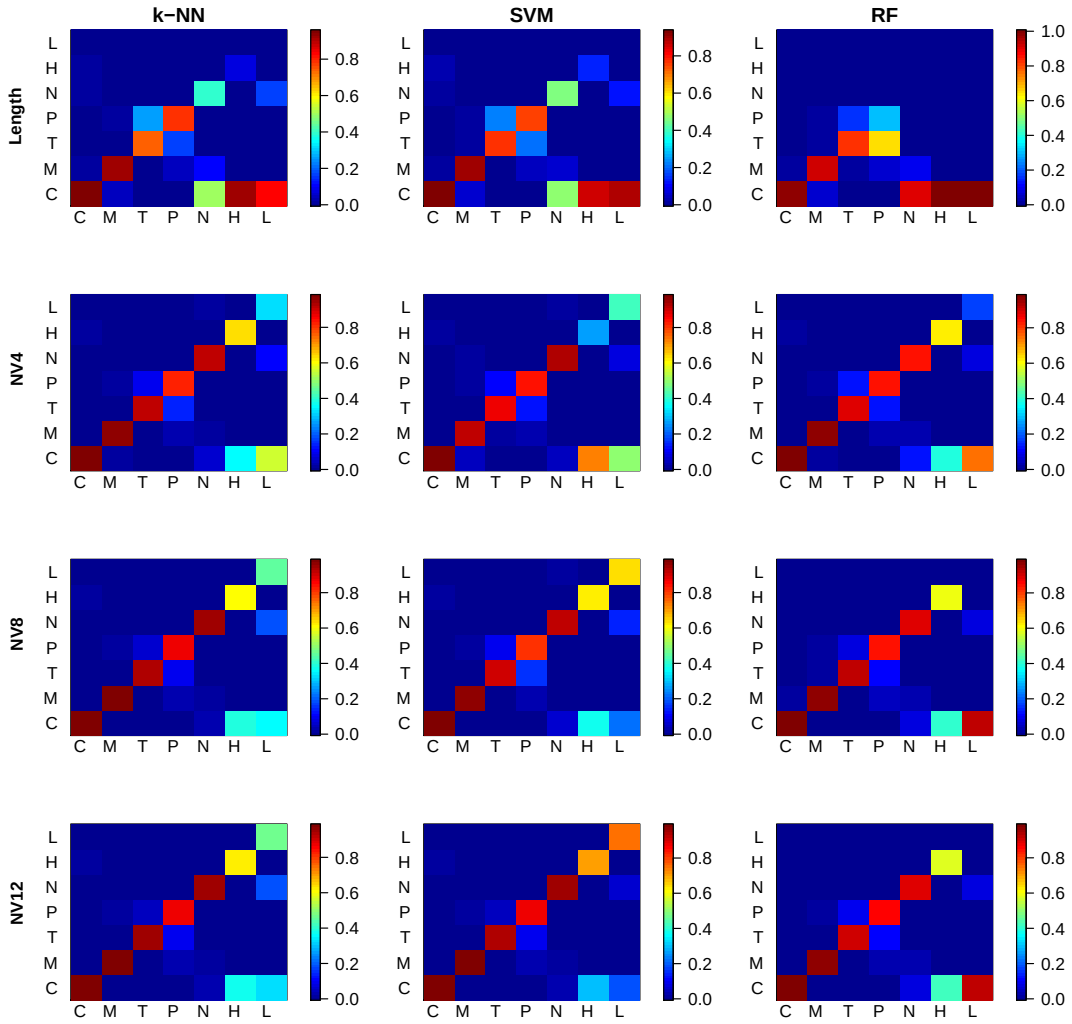


Figure 6.3: Confusion matrix for the ICTV Order experiment.
In each subplot, the x-axis represents the true class labels and y-axis represents the prediction. Colours represent the percentage of viruses from class x being predicted as a member of class y. Diagonal entries represent rates of correct classification, where x equals y; off-diagonal entries are rates of misclassification. Classes are ordered in decreasing size from left to right and bottom to top.

should be one of the objectives when designing new classification methodologies. Hence, the second aim here is to construct a new classifier that improves classification performance by reducing misclassification of difficult viruses.

Experiments in this section use all labelled and non-segmented viruses in the dataset, which gives 3,625 for Baltimore Class experiments and 1,865 for ICTV Order experiments (Table 2.1). NV12 is used as feature for a virus genome sequence

since it gives the best performance in previous experiments. As in the previous sections, k-NN, SVM and RF are the classifiers.

To predict the class of a virus, we use a leave-one-out strategy: the testing set is a singleton set consisting of only this virus, while the training set contains all the others. We train the classifiers and optimize their parameters using the training set, and then predict the class label for the virus in the testing set using the trained classifiers. This procedure is repeated until every virus has been used as the test sample. Hence, each virus is associated with two class labels, one predicted from the classifiers and another obtained from the dataset. To assess the performance of a classifier, we compute the error rate as the ratio between the number of viruses it misclassifies and the total number of viruses. To assess the difficulty of a virus, we repeat the above procedure for each of the three classifiers and count the number of classifiers that have misclassified the given virus: difficulty levels 3, 2, 1 and 0 are for viruses misclassified by 3, 2, 1 and 0 classifiers respectively.

6.3.2 Difficult viruses are from class boundaries

Table 6.3 shows the number of viruses with different levels of difficulty and Fig. 6.4 shows the distribution of difficulty levels. Comparing Fig. 6.4 with Fig. 5.9, we observe that higher errors are at boundaries where there is a large degree of overlapping between different classes, and low errors are in the regions consisting of a single class. We summarize the membership of level 3 viruses in Table 6.4 and 6.5. Among all the 120 level 3 viruses identified from the Baltimore Class experiments, 104 are currently unable to be placed into an ICTV Order by the NCBI.

Schemes	**Level 3**	**Level 2**	**Level 1**	**Level 0**
Baltimore Classes	120	105	162	3238
ICTV Orders	31	33	80	1721

Table 6.3: Number of viruses with different levels of difficulty.

Difficulty levels 3, 2, 1 and 0 for viruses misclassified by 3, 2, 1 and 0 classifiers respectively.

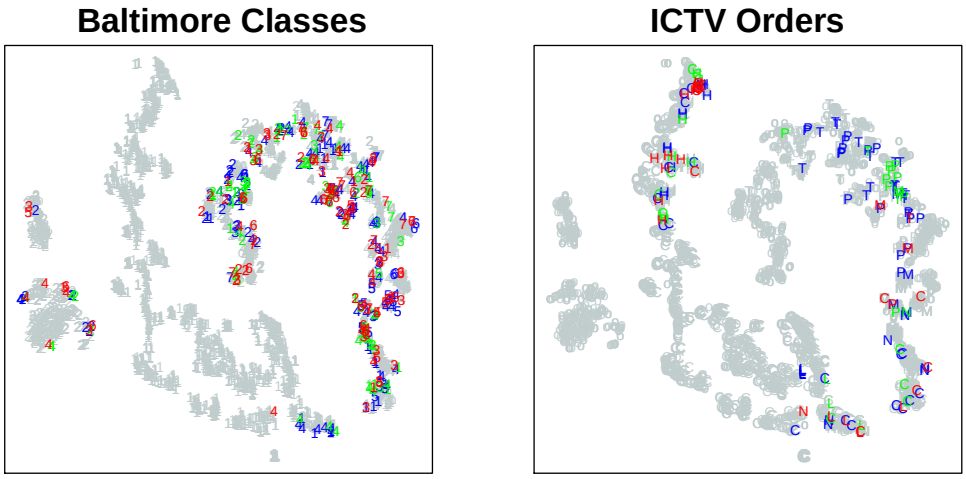


Figure 6.4: t-SNE visualisation of the distribution of difficulty levels of viruses. The colour grey represents unlabelled and level 0 viruses; blue, green and red are for level 1, 2 and 3 viruses respectively, according to results of the Baltimore Class experiments (left) and ICTV Order experiments (right). Point styles for unlabelled viruses are circles and for labelled viruses abbreviations of their class names (Arabic numbers 1-7 are used for the seven Baltimore Classes instead of the Roman numerals I-VII).

	I	II	III	IV	V	VI	VII
C	1	0	0	0	0	0	0
H	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0
M	0	0	0	0	9	0	0
N	0	0	0	2	0	0	0
P	0	0	0	2	0	0	0
T	0	0	0	2	0	0	0
Unlabelled	6	17	26	13	1	25	16

Table 6.4: Membership of level 3 viruses identified in Baltimore Class experiments.

Table entries are the numbers of viruses whose Baltimore Classes are misclassified by all three classifiers.

6.3.3 A combined classifier improves classification performance

From Table 6.3, we observe that the majority of viruses can be correctly classified by all classifiers and the likelihood of a virus being misclassified by more than one classifier is relatively low. This motivates us to construct a new classifier, MV (Majority Vote), which combines the predictions of the three classifiers to improve

	C	H	L	M	N	P	T
I	11	11	4	0	0	0	0
II	0	0	0	0	0	0	0
III	0	0	0	0	0	0	0
IV	0	0	0	0	1	1	1
V	0	0	0	2	0	0	0
VI	0	0	0	0	0	0	0
VII	0	0	0	0	0	0	0

Table 6.5: Membership of level 3 viruses identified in ICTV Orders experiments.

Table entries are the numbers of viruses whose ICTV Orders are misclassified by all three classifiers.

performance on difficult viruses. MV predicts with the majority votes from the three classifiers, and takes the SVM prediction when all of them disagree since SVM gives the lowest mean error rates according to Table 6.2. An intuitive justification for this choice and its ability to reduce misclassification of difficult viruses is as follows. When misclassification occurs among the three classifiers, MV will give the correct prediction if one classifier makes a mistake, or two classifiers make different mistakes but SVM is correct; MV gives the incorrect prediction only when two classifiers make the same mistake, or two classifiers make different mistakes and SVM is one of them, or all three classifiers make a mistake. This means MV will be able to correct all level 1 viruses and certain portion of level 2 viruses. However, it is unable to improve any level 3 viruses.

Experimental results confirm that MV tends to outperform any individual classifiers (Table 6.6). SVM is the best among the three classifiers but MV improves performance further. In terms of error rates of individual classes, the best is either SVM or MV: SVM is the best for 5 Baltimore Classes and 4 ICTV Orders, and MV is the best for 2 Baltimore Classes and 4 ICTV Orders. However, MV is more consistent - it is always no worse than the second best, and hence has lower total error rates. On the other hand, the inadequacy of MV is also apparent. First, MV does not improve problems caused by imbalanced class sizes. Table 6.6 shows a similar correlation between the class size and the error rates as in Fig. 6.2 and 6.3: smaller classes tend to give higher errors. Second, MV is unable to give an improvement

for viruses where two or three classifiers make the same mistake. However, the chances that the same mistake is made for a level 2 or 3 virus tends to be high: 67.6% of level 2 viruses in Baltimore Class experiments and 90.9% in ICTV Order experiments make the same incorrect prediction; 65.8% of level 3 viruses in Baltimore Class experiments and 87.1% in ICTV Order experiments make the same incorrect prediction. Level 3 viruses are difficult to predict by any classifiers in our experiments, which is likely caused by insufficient representation when using NV.

Schemes	**Classifiers**	**Classes**							**Total**
Baltimore Class		I	IV	II	V	VII	III	VI	
	k-NN	0.016	0.071	0.079	0.122	0.271	0.446	0.711	0.067
	SVM	**0.009**	0.078	**0.059**	0.154	**0.219**	**0.398**	**0.684**	0.060
	RF	0.020	0.062	0.081	0.128	0.333	0.614	0.921	0.075
	MV	**0.009**	**0.053**	0.067	**0.109**	0.240	0.434	0.711	**0.057**
ICTV Orders		C	M	T	P	N	H	L	
	k-NN	0.020	0.032	0.041	0.111	0.030	0.303	0.538	0.043
	SVM	0.017	**0.013**	0.061	0.081	**0.015**	**0.242**	**0.308**	0.035
	RF	0.018	0.032	0.061	0.104	0.119	0.318	1.000	0.050
	MV	**0.016**	**0.013**	**0.020**	**0.074**	0.030	0.258	0.462	**0.033**

Table 6.6: Classification error rates of different classifiers.

The Table shows classification error rates of each class using the classifiers k-NN, SVM, RF and MV. Classes in each scheme are ordered by decreasing size. The column "Total" shows the total error rate over all classes. The lowest error rates of each class are highlighted in bold.

6.4 Predicting the Baltimore Class and ICTV Order for currently unlabelled viruses

6.4.1 Design

This section aims to predict the Baltimore Class and/or ICTV Order for currently unlabelled non-segmented viruses using the best combination of feature and classifier identified in earlier experiments.

All the 3,699 non-segmented viruses (Table 2.1) are used. The training set consists of all the labelled ones (3,625 for Baltimore Class experiments and 1,865

for ICTV Order experiments), and the testing set consists of all the unlabelled ones (74 for Baltimore Class experiments and 1,834 for ICTV Order experiments). We train classifiers and optimise their parameters using viruses in the training set, and then predict classes for these in the testing set.

6.4.2 Prediction of unlabelled viruses

This section shows predictions of the Baltimore Class or/and ICTV Order for viruses in the dataset currently unlabelled by the schemes. Table 6.7 summarizes the predicted membership for all currently unlabelled viruses in the dataset and Table 6.8 shows agreement between the three classifiers in predicting the class of a virus.

Schemes	**Predicted classes**						
Baltimore Class	I	II	III	IV	V	VI	VII
	37	27	1	9	0	0	0
ICTV Orders	C	H	L	M	N	P	T
	716	22	9	252	27	369	439

Table 6.7: Predicted classes for unlabelled viruses.

The Table shows the number of currently unlabelled viruses being predicted in each class.

Schemes	**All agree**	**Two agree**	**None agree**
Baltimore Class	65	7	2
ICTV Orders	916	821	97

Table 6.8: Prediction agreement between classifiers.

The Table shows the number of viruses whose predicted classes are agreed by three, two and zero classifiers.

6.5 Predicting the host of non-segmented viruses

6.5.1 Design

This section aims to study the predictive power of the nucleotide statistics of virus genome sequences in classifying their host groups. The dataset used in this section

contains the 3,644 host-annotated viruses in Table 2.4.

6.5.2 Virus host prediction

Our results (Table 6.9) show that virus hosts can be predicted using features derived only from the genome sequence, even though the error rates are higher than those in the Baltimore Class or ICTV Order experiments. This is interesting because the host of a virus is its phenotype, and such taxonomy does not involve genome sequence as part of the criteria. Being able to predict hosts using genome sequence-based methods potentially gives new perspectives on genome sequence-based alignment-free virus taxonomy as seemingly irrelevant biological properties are actually intrinsically encoded in the sequences. It can also significantly advance virology research if many useful biological properties can be derived from genome sequences.

Feature	k-NN	SVM	RF
Length	**0.342 ± 0.0128**	0.375 ± 0.041	0.436 ± 0.026
NV4	**0.100 ± 0.010**	0.109 ± 0.008	0.127 ± 0.012
NV8	**0.098 ± 0.009**	0.102 ± 0.010	0.115 ± 0.015
NV12	**0.094 ± 0.006**	0.098 ± 0.008	0.109 ± 0.015

Table 6.9: Error rates for virus host prediction.

6.6 Summary

In this chapter, we studied the predictive powers of features based on nucleotide statistics.

We provided a systematic experimental framework for applying machine learning techniques to virus taxonomy. We investigated the performance of various combinations of features and classifiers, from simple k-NN to more sophisticated SVM and RF. We optimised their performance using variable normalisation and parameter tuning, and avoided performance bias through repeated experiments with randomised training and testing samples.

The experimental results agree with the qualitative observations from the visualisation results. We found that Length can give respectable classification performance but a similar Length distribution is one of the confounding factors causing

high error rates for small classes. NV4 can significantly improve performance from Length, but NV8 and NV12 only give a marginal further improvement in terms of the overall error rates. However, it is evident that more sophisticated features improve performance for small classes.

Realising the difficulty of predicting sequences from minority classes and class boundaries, we formulated a majority voting scheme by combining the predictions from competing classifiers. We observed improved overall performance, yet certain sequences are still consistently misclassified by all the classifiers.

In the next chapter we will extend this chapter to multi-class classification using word and compression-based features, and using the best experimental settings identified, predict labels for currently unlabelled sequences and extend our study to predicting hosts.

Chapter 7

Multi-class classification using word and compression-based features

This chapter extends the previous chapter on features based on single nucleotides to features derived from *k*-mers and the entire sequence. The aim is to investigate the predictive power of a wide range of word- and compression-based features for genome sequences and classifiers in the task of virus taxonomy.

Similarly to the previous chapters, we start by predicting the Baltimore Classes and ICTV Orders for non-segmented viruses using different combinations of features and classifiers. Next, using the best experimental settings we have, we identify sequences that are difficult to correctly classify, predict labels for currently unlabelled sequences as well as predicting virus host groups. We then extend the study to predict class labels for multi-segmented viruses.

7.1 Methods

Studies in this chapter use the same experimental framework as in the last chapter. The classifiers used here include k-NN, Linear-SVM, Radial-SVM and L1-SVM. Table 7.1 summarises the range of parameters tuned during cross-validation.

Classifier	Range of parameter values
k-NN	$k \in \{1,2,3,...,10\}$
Linear-SVM	$C \in \{2^5,2^6,2^7,...,2^{17}\}$
Radial-SVM	$C \in \{2^5,2^6,2^7,...,2^{17}\}$ $\gamma \in \{2^{-18},2^{-16},2^{-14},...,2^{13}\}$
L1-SVM	$\lambda \in \{2^{-4},2^{-3},2^{-2},...,2^8\}$

Table 7.1: Range of parameter values used during cross-validation.

7.2 Features based on *k*-mer statistics

7.2.1 Design

In this section, we assess classification performance using different types of *k*-mer related features in the task of virus taxonomy, including *k*-mer counts, *k*-mer NV and *k*-mer RTD.

7.2.2 *k*-mer NV improves performance from NV12

Features based on *k*-mers are rich representations of the original sequences, which can have very high dimension (see Table 4.3 for summary of *k*-mer features). Fig. 5.11 shows a t-SNE visualisation of the sequences in the dataset using a high dimensional 6-mer NV feature.

Table 7.2 shows the overall classification error rates using different classifiers and *k*-mer NV with different *k*, where 1-mer NV is the same as NV12 in Chapter 6. We can observe from the table that with suitable *k*-mers, performance improves from 1-mer NV for all the classifiers in both the Baltimore Class and ICTV Order experiments. However, longer *k*-mers do not always improve performance and may add noise during classification. In fact, as *k* increases, error rates follow the same trend in that they first decrease and then increase. In Baltimore Class experiments, the best features for the k-NN classifier, Linear-SVM and Radial-SVM are 2-mer NV, 5-mer NV and 4-mer NV respectively, with the overall best being the combination of 4-mer NV and Radial-SVM; in ICTV Order experiments, the best for the three classifiers are 2-mer NV, 4-mer NV and 4-mer NV respectively, with the overall best being 4-mer NV and SVM (with both linear and radial kernel).

In addition, different classifiers favour different levels of feature complexity.

For instance, k-NN prefers relatively simpler features and does not perform well with complex ones. It is sometimes the best classifier when using simple features, such as Length, NV1, NV4 and NV8 (see Table 6.2), but with complex features, such as 4-mer NV, 5-mer NV and 6-mer NV, the performance can be even worse than 1-mer NV. In contrast, Linear-SVM favours complex features. It is the worst with 1-mer NV, but can be the best with 3-mer NV and more complex ones in ICTV Order experiments. Radial-SVM can perform well with both simple and complex features. It is the best with all the features in Table 7.2 and half of the features in Table 6.2.

Baltimore Classes			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	0.076 ± 0.006	0.170 ± 0.011	$\mathbf{0.067 \pm 0.008}$
2-mer	0.059 ± 0.006	0.078 ± 0.008	$\mathbf{0.035 \pm 0.008}$
3-mer	0.069 ± 0.007	0.052 ± 0.005	$\mathbf{0.029 \pm 0.008}$
4-mer	0.091 ± 0.008	0.043 ± 0.005	$\mathbf{0.028 \pm 0.005}$
5-mer	0.136 ± 0.010	0.039 ± 0.006	$\mathbf{0.032 \pm 0.006}$
6-mer	0.142 ± 0.011	0.043 ± 0.008	$\mathbf{0.037 \pm 0.006}$

ICTV Orders			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	0.049 ± 0.007	0.049 ± 0.008	$\mathbf{0.035 \pm 0.007}$
2-mer	0.036 ± 0.006	0.026 ± 0.006	$\mathbf{0.021 \pm 0.003}$
3-mer	0.041 ± 0.006	$\mathbf{0.009 \pm 0.004}$	$\mathbf{0.009 \pm 0.003}$
4-mer	0.052 ± 0.009	$\mathbf{0.007 \pm 0.003}$	$\mathbf{0.007 \pm 0.002}$
5-mer	0.067 ± 0.011	$\mathbf{0.008 \pm 0.003}$	$\mathbf{0.008 \pm 0.003}$
6-mer	0.071 ± 0.013	$\mathbf{0.009 \pm 0.004}$	$\mathbf{0.009 \pm 0.003}$

Table 7.2: Classification error rate of different features and classifiers using the k-mer NV of ACGT.

The table shows the mean and standard deviation of classification error rates using the k-mer NV. The lowest mean error rates in each row are highlighted in bold.

7.2.3 Concatenating statistics of subwords of a k-mer does not improve performance over k-mer NV

A rich feature representation can usually improve classification performance, but there is a trade-off between better performance and computational efficiency. Table 7.2 shows that with a suitable k, k-mer NV improves performance from 1-mer NV. Here, we study whether concatenating the statistics of subwords of a k-mer (k'-mers for any $k' < k$) to k-mer NV can improve performance. Our results (Table 7.3) show that a concatenated k-mer NV does not improve performance over k-mer NV (Table 7.2). In fact, the error rates are almost the same for each combination of features and classifiers. This suggests that the statistics of subwords become redundant once the k-mer NV is given. To validate this, we apply L1-SVM to the concatenated k-mer NV. Analysing weights w learned from L1-SVM experiments, we find that most non-zero weights are associated with words of length k rather than shorter ones, confirming the unimportance of the statistics of subwords of k-mers.

7.2.4 k-mer counts perform no worse than k-mer NV

By analysing L1-SVM weights, we also find the importance of counts in k-mer NV. For Baltimore Class experiments (Fig. 7.1), the majority counts are selected together with a large number of mean positions and normalised variances; but for ICTV Order experiments (Fig. 7.2), the selected variables are mainly counts. This indicates that for ICTV Orders, counts play a key part in classification while mean position and variance are far less important. For Baltimore Classes, counts are still important, but in contrast to ICTV Orders, their degree of importance is weaker as the problem is intrinsically more difficult. The predictive power of k-mer counts are confirmed in Table 7.4, where k-mer counts perform as well as the full k-mer NV (Table 7.2) in the Baltimore Class experiments and slightly better in the ICTV Order experiments. Such results are positive because feature dimension can be reduced by more than a third from the concatenated k-mer NV to the simpler k-mer counts.

Baltimore Classes			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	0.076 ± 0.006	0.170 ± 0.011	$\mathbf{0.069 \pm 0.008}$
2-mer	0.059 ± 0.006	0.078 ± 0.008	$\mathbf{0.035 \pm 0.008}$
3-mer	0.069 ± 0.007	0.052 ± 0.005	$\mathbf{0.028 \pm 0.006}$
4-mer	0.092 ± 0.008	0.043 ± 0.005	$\mathbf{0.028 \pm 0.005}$
5-mer	0.135 ± 0.011	0.039 ± 0.006	$\mathbf{0.032 \pm 0.005}$
6-mer	0.142 ± 0.011	0.043 ± 0.008	$\mathbf{0.037 \pm 0.006}$

ICTV Orders			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	0.049 ± 0.007	0.049 ± 0.008	$\mathbf{0.037 \pm 0.009}$
2-mer	0.036 ± 0.006	0.026 ± 0.006	$\mathbf{0.021 \pm 0.006}$
3-mer	0.041 ± 0.006	$\mathbf{0.008 \pm 0.004}$	$\mathbf{0.008 \pm 0.004}$
4-mer	0.052 ± 0.009	$\mathbf{0.007 \pm 0.003}$	$\mathbf{0.007 \pm 0.004}$
5-mer	0.067 ± 0.011	$\mathbf{0.008 \pm 0.003}$	$\mathbf{0.008 \pm 0.005}$
6-mer	0.071 ± 0.013	$\mathbf{0.009 \pm 0.004}$	$\mathbf{0.009 \pm 0.003}$

Table 7.3: Classification error rate of different features and classifiers using the concatenated k-mer NV of ACGT.

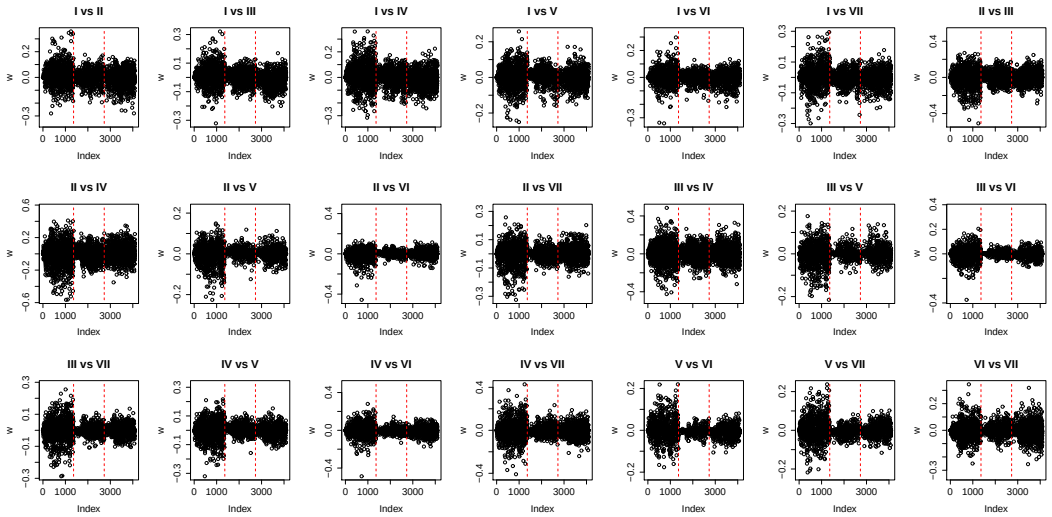


Figure 7.1: Scatter plots of L1-SVM weights w learned from each pair of Baltimore Classes during a one-vs-one multi-class scheme.

The Figure shows w values learned using concatenated 5-mer NV and $\lambda = 2^5$, as early results show that this combination performs the best for L1-SVM in the Baltimore Class experiments. Black circles represent w associated with feature variables of the concatenated 5-mer NV. Points further away from zero indicate values with larger magnitudes, hence higher importance of the associated variables during classification. Red vertical lines separate count (left), mean position (middle) and normalised variance (right).

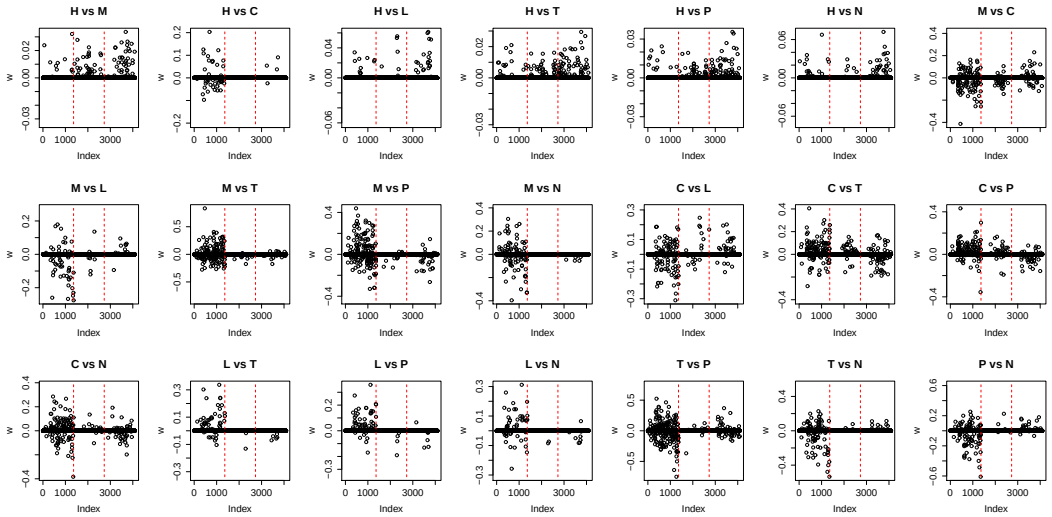


Figure 7.2: Scatter plots of L1-SVM weights w learned from each pair of ICTV Orders during a one-vs-one multi-class scheme.

This is the same as Fig. 7.1 but for ICTV Order experiments where the best combination is the concatenated 5-mer NV and $\lambda = 2$.

Baltimore Classes			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	0.089 ± 0.008	0.228 ± 0.012	$\mathbf{0.086 \pm 0.005}$
2-mer	$\mathbf{0.046 \pm 0.006}$	0.130 ± 0.005	0.048 ± 0.006
3-mer	$\mathbf{0.034 \pm 0.006}$	0.071 ± 0.007	0.036 ± 0.008
4-mer	0.029 ± 0.006	0.054 ± 0.006	$\mathbf{0.027 \pm 0.006}$
5-mer	$\mathbf{0.030 \pm 0.006}$	0.044 ± 0.008	0.032 ± 0.006
6-mer	0.053 ± 0.006	0.042 ± 0.007	$\mathbf{0.035 \pm 0.006}$

ICTV Orders			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	$\mathbf{0.058 \pm 0.011}$	0.136 ± 0.013	0.095 ± 0.010
2-mer	$\mathbf{0.025 \pm 0.005}$	0.042 ± 0.006	0.028 ± 0.007
3-mer	0.016 ± 0.004	0.009 ± 0.003	$\mathbf{0.008 \pm 0.004}$
4-mer	0.013 ± 0.004	$\mathbf{0.006 \pm 0.002}$	$\mathbf{0.006 \pm 0.002}$
5-mer	0.012 ± 0.004	$\mathbf{0.006 \pm 0.003}$	$\mathbf{0.006 \pm 0.003}$
6-mer	0.016 ± 0.005	$\mathbf{0.007 \pm 0.002}$	$\mathbf{0.007 \pm 0.002}$

Table 7.4: Classification error rate of different features and classifiers using k-mer counts of ACGT.

7.2.5 Richer alphabets improve classification performance

Here, we study the influence of alphabets of k-mers (see Table 4.2 for alphabets) on classification performance. Our results show that features using a richer alphabet outperform those using simpler ones: the four-letter alphabet (ACGT) (Table 7.4) performs better than the concatenated three two-letter alphabets (SW-RY-MK) (Table 7.8), which are better than the individual two-letter alphabets (SW, RY, MK) (Table 7.5, 7.6, 7.7), which in turn are better than the single-letter alphabet (Length) (Table 6.2).

Baltimore Classes			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	**0.154 ± 0.015**	0.237 ± 0.012	**0.154 ± 0.013**
2-mer	0.125 ± 0.008	0.220 ± 0.011	**0.121 ± 0.011**
3-mer	0.097 ± 0.006	0.203 ± 0.010	**0.082 ± 0.007**
4-mer	0.092 ± 0.007	0.184 ± 0.012	**0.082 ± 0.008**
5-mer	0.089 ± 0.007	0.168 ± 0.010	**0.077 ± 0.008**
6-mer	0.088 ± 0.010	0.162 ± 0.008	**0.081 ± 0.009**

ICTV Orders			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	**0.090 ± 0.015**	0.152 ± 0.017	0.099 ± 0.010
2-mer	**0.068 ± 0.013**	0.134 ± 0.008	0.087 ± 0.009
3-mer	**0.066 ± 0.008**	0.109 ± 0.014	0.075 ± 0.012
4-mer	**0.058 ± 0.013**	0.094 ± 0.008	0.071 ± 0.013
5-mer	**0.053 ± 0.012**	0.071 ± 0.011	0.056 ± 0.007
6-mer	**0.052 ± 0.012**	0.065 ± 0.012	0.055 ± 0.006

Table 7.5: Classification error rate of different features and classifiers using k-mer counts of SW.

Baltimore Classes			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	**0.140 ± 0.010**	0.243 ± 0.012	0.141 ± 0.008
2-mer	**0.113 ± 0.005**	0.233 ± 0.008	0.117 ± 0.005
3-mer	0.094 ± 0.007	0.204 ± 0.010	**0.093 ± 0.006**
4-mer	0.082 ± 0.005	0.180 ± 0.007	**0.080 ± 0.006**
5-mer	0.074 ± 0.005	0.167 ± 0.010	**0.069 ± 0.006**
6-mer	0.071 ± 0.006	0.141 ± 0.009	**0.064 ± 0.006**

ICTV Orders			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	0.091 ± 0.013	0.150 ± 0.009	**0.085 ± 0.010**
2-mer	**0.081 ± 0.009**	0.128 ± 0.008	0.082 ± 0.009
3-mer	**0.050 ± 0.011**	0.100 ± 0.008	0.051 ± 0.010
4-mer	0.039 ± 0.007	0.062 ± 0.009	**0.031 ± 0.005**
5-mer	0.035 ± 0.006	0.037 ± 0.006	**0.033 ± 0.007**
6-mer	0.034 ± 0.007	0.035 ± 0.006	**0.030 ± 0.006**

Table 7.6: Classification error rate of different features and classifiers using k-mer counts of RY.

Baltimore Classes			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	0.134 ± 0.009	0.241 ± 0.013	$\mathbf{0.125 \pm 0.011}$
2-mer	$\mathbf{0.115 \pm 0.007}$	0.246 ± 0.013	0.118 ± 0.009
3-mer	0.108 ± 0.008	0.232 ± 0.011	$\mathbf{0.101 \pm 0.005}$
4-mer	0.102 ± 0.009	0.213 ± 0.012	$\mathbf{0.100 \pm 0.006}$
5-mer	0.099 ± 0.008	0.203 ± 0.012	$\mathbf{0.090 \pm 0.008}$
6-mer	0.096 ± 0.007	0.196 ± 0.014	$\mathbf{0.091 \pm 0.009}$

ICTV Orders			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	$\mathbf{0.103 \pm 0.008}$	0.139 ± 0.012	0.107 ± 0.012
2-mer	$\mathbf{0.081 \pm 0.015}$	0.125 ± 0.015	0.083 ± 0.010
3-mer	$\mathbf{0.068 \pm 0.009}$	0.112 ± 0.015	0.074 ± 0.013
4-mer	0.053 ± 0.006	0.077 ± 0.014	$\mathbf{0.052 \pm 0.011}$
5-mer	0.049 ± 0.006	0.068 ± 0.011	$\mathbf{0.047 \pm 0.009}$
6-mer	0.045 ± 0.007	0.058 ± 0.007	$\mathbf{0.044 \pm 0.008}$

Table 7.7: Classification error rate of different features and classifiers using k-mer counts of MK.

Baltimore Classes			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	0.094 ± 0.007	0.228 ± 0.012	$\mathbf{0.088 \pm 0.008}$
2-mer	0.064 ± 0.008	0.169 ± 0.008	$\mathbf{0.059 \pm 0.006}$
3-mer	0.048 ± 0.008	0.135 ± 0.010	$\mathbf{0.047 \pm 0.007}$
4-mer	0.045 ± 0.008	0.115 ± 0.005	$\mathbf{0.043 \pm 0.005}$
5-mer	0.043 ± 0.008	0.099 ± 0.007	$\mathbf{0.040 \pm 0.007}$
6-mer	0.041 ± 0.008	0.083 ± 0.006	$\mathbf{0.038 \pm 0.006}$

ICTV Orders			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	0.071 ± 0.012	0.134 ± 0.013	0.100 ± 0.009
2-mer	$\mathbf{0.041 \pm 0.008}$	0.070 ± 0.004	0.044 ± 0.005
3-mer	0.026 ± 0.005	0.035 ± 0.004	$\mathbf{0.024 \pm 0.005}$
4-mer	0.023 ± 0.003	0.021 ± 0.005	$\mathbf{0.018 \pm 0.004}$
5-mer	0.019 ± 0.003	0.019 ± 0.003	$\mathbf{0.017 \pm 0.003}$
6-mer	0.019 ± 0.004	0.020 ± 0.004	$\mathbf{0.016 \pm 0.006}$

Table 7.8: Classification error rate of different features and classifiers using concatenated k-mer counts of SW, RY and MK.

7.2.6 *k*-mer counts outperform RTD

The results show that *k*-mer counts (Table 7.4) outperform RTD (Table 7.9). One of the reasons RTD does not perform well may be that it has lost information of the genome length, whose importance was demonstrated in the previous chapter. Therefore, we construct a new feature by concatenating *k*-mer counts and RTD to include information of the genome length. Table 7.10 shows that the new feature improves performance over RTD (Table 7.9) but is still worse than *k*-mer counts (Table 7.4).

Baltimore Classes			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	0.229 ± 0.013	0.354 ± 0.020	$\mathbf{0.169 \pm 0.007}$
2-mer	0.124 ± 0.011	0.212 ± 0.013	$\mathbf{0.078 \pm 0.009}$
3-mer	0.126 ± 0.008	0.126 ± 0.009	$\mathbf{0.060 \pm 0.005}$
4-mer	0.232 ± 0.014	0.093 ± 0.007	$\mathbf{0.063 \pm 0.005}$
5-mer	0.306 ± 0.019	0.097 ± 0.009	$\mathbf{0.080 \pm 0.009}$
6-mer	0.344 ± 0.018	0.102 ± 0.013	$\mathbf{0.078 \pm 0.008}$

ICTV Orders			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	0.137 ± 0.016	0.186 ± 0.015	$\mathbf{0.123 \pm 0.012}$
2-mer	0.060 ± 0.007	0.074 ± 0.012	$\mathbf{0.037 \pm 0.005}$
3-mer	0.061 ± 0.009	0.024 ± 0.005	$\mathbf{0.014 \pm 0.004}$
4-mer	0.102 ± 0.019	0.020 ± 0.006	$\mathbf{0.016 \pm 0.005}$
5-mer	0.292 ± 0.025	0.017 ± 0.004	$\mathbf{0.016 \pm 0.004}$
6-mer	0.484 ± 0.008	0.020 ± 0.005	$\mathbf{0.019 \pm 0.004}$

Table 7.9: Classification error rate of different features and classifiers using RTD of ACGT.

Baltimore Classes			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	**0.089 ± 0.008**	0.155 ± 0.010	0.106 ± 0.012
2-mer	**0.045 ± 0.006**	0.087 ± 0.007	0.056 ± 0.010
3-mer	0.048 ± 0.004	0.074 ± 0.005	**0.047 ± 0.005**
4-mer	0.184 ± 0.011	0.081 ± 0.007	**0.058 ± 0.006**
5-mer	0.303 ± 0.017	0.090 ± 0.008	**0.079 ± 0.006**
6-mer	0.345 ± 0.018	0.077 ± 0.009	**0.068 ± 0.006**

ICTV Orders			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	0.058 ± 0.011	0.057 ± 0.009	**0.048 ± 0.012**
2-mer	0.025 ± 0.005	0.029 ± 0.010	**0.018 ± 0.006**
3-mer	0.019 ± 0.005	0.014 ± 0.004	**0.010 ± 0.004**
4-mer	0.071 ± 0.014	0.010 ± 0.004	**0.009 ± 0.004**
5-mer	0.288 ± 0.024	0.014 ± 0.004	**0.013 ± 0.004**
6-mer	0.484 ± 0.008	0.015 ± 0.005	**0.015 ± 0.005**

Table 7.10: Classification error rate of different features and classifiers using the concatenation of count and RTD of ACGT.

7.3 Features based on absent words

7.3.1 Design

In this section, we study the performance of features based on absent words, i.e. k-mers that are possible but missing in the sequences, focusing on MAW.

We used the algorithm and code described in [145] to compute the set of MAW of each genome sequence. We extract MAW with length ranges from 1 to 1,000, which is much larger than the majority MAW in a sequence. The benefits of doing so are two-fold. First, we would like to extract as many MAW as possible for a reasonable computational cost. Second, setting a large maximum length is likely to identify the longest MAW, which are shown to be more distinguishable for different sequences than short ones [118].

In addition, we also compute the MAW of each sequence with its Reverse Complement (RC) concatenated, and compare this to the ones without (noRC). The potential benefit of RC is that it considers words that might occur in the reverse complement strand but be absent from the direct strand. For example, given a sequence ACCGTA, the input sequence for extracting MAW in a noRC setting is the original one, ACCGTA; in the RC setting, the original sequence ACCGTA is concatenated with its reverse complement TACGGT, and the input sequence is ACCGTA$TACGGT (the $ sign is used to flag artificial words formed in the boundary and any MAW containing it will be removed).

7.3.2 MAW performs well

Table 7.11 shows that MAW can give respectable performance in both Baltimore Class and ICTV Order experiments. Classification errors for ICTV Orders are lower than those for Baltimore Classes, which is consistent with the results from other features in our study. However, the best performing combination of feature and difference measure is noRC and JD, which is actually the opposite of the results in [121], where RC outperforms noRC and $LWI_{\cap}$ outperforms JD. The main cause of this inconsistency could be the datasets used. Experiments in [121] use a small dataset of the first axon sequences of β-globin genes of 11 species, whose lengths

vary from 86 to 105 base pairs (for details of the dataset see [146]). In contrast, the dataset we use is much larger, with sequence length varying significantly from 859 to 2,473,870 base pairs. RC can be redundant given the long sequences in our dataset, and the significant variation in sequence length tends to bias the $LWI_{\cap}$ measure. Typically, the intersection between two long genome sequences tends to contain more elements than between two short ones, hence long sequences give a smaller $LWI_{\cap}$, suggesting a lower level of difference. However, JD can alleviate this problem by using the ratio between $|MAW_{S_1} \cap MAW_{S_2}|$ and $|MAW_{S_1} \cup MAW_{S_2}|$ (see Section 4.2.6 for difference measures).

Baltimore Classes		
Feature	**Difference Measure**	
	$LWI_{\cap}$	JD
noRC	0.193 ± 0.007	$\mathbf{0.027 \pm 0.006}$
RC	0.228 ± 0.014	0.030 ± 0.004

ICTV Orders		
Feature	**Difference Measure**	
	$LWI_{\cap}$	JD
noRC	0.159 ± 0.010	$\mathbf{0.014 \pm 0.005}$
RC	0.193 ± 0.012	0.017 ± 0.006

Table 7.11: Classification error rate of different features and difference measures using MAW.

7.4 Features based on compression

7.4.1 Design

The purpose of this section is to study the classification performance of predicting Baltimore Classes and ICTV Orders using features derived from compression methods. The assumption for this study is that similar sequences contain similar patterns and tend to have similar compression ratios for a given tool. Hence, the features will be representative of a sequence and the distance between features reflects the distance between the original sequences.

Features for a genome sequence consist of compression ratios obtained using different compression tools. We first consider genome sequences as a piece of text and compress them in a regular way using general-purpose compression tools: bzip2, gzip, xz, zip. Then, we use features derived using reference-free DNA-specific compression tools: DELIMINATE [133], MFCompress [134] and LEON [136]. All the features are summarised in Table 4.4. For each tool, the parameters are set to achieve the best compression ratio.

7.4.2 General-purpose compression

The distribution of compression ratios from each tool is shown in Fig. 5.19 and 5.21. We construct features by combining the compression ratios of bzip2, gzip, xz and zip (CRGP). Since bzip2 gives the best compression performance, we explore two other features related to its ratio. One is to use its ratio as a single variable feature (CRB), and the other is a 2D feature that combines its ratio with log transformed genome length (CRBL). For details of features, see Table 4.4.

The classification performance is shown in Table 7.12. The best performance for predicting Baltimore Classes and ICTV Orders are both achieved using the feature CRGP, with error rates of 0.139 and 0.092 respectively.

Baltimore Classes			
Classifier	CRGP	CRB	CRBL
kNN	0.154 ± 0.008	$\mathbf{0.228 \pm 0.009}$	0.187 ± 0.014
SVM	$\mathbf{0.139 \pm 0.006}$	0.229 ± 0.009	$\mathbf{0.177 \pm 0.013}$

ICTV Orders			
Classifier	CRGP	CRB	CRBL
kNN	0.101 ± 0.011	$\mathbf{0.183 \pm 0.007}$	0.139 ± 0.012
SVM	$\mathbf{0.092 \pm 0.007}$	$\mathbf{0.183 \pm 0.009}$	$\mathbf{0.117 \pm 0.011}$

Table 7.12: Classification performance using features based on compression ratios of general-purpose compression tools.

7.4.3 DNA-specific compression

The distribution of compression ratios from each tool is shown in Fig. 5.20 and 5.22. We construct features by combining the compression ratios of DELIMINATE, MFCompress and LEON (CRDNA). Since LEON gives the best compression performance, we explore two other features related to its ratio. One is to use its ratio as a single variable feature (CRL), and the other combines its ratio with log transformed genome length (CRLL). In addition, we also construct a feature that combines the ratios of both general-purpose and DNA-specific tools (CRA). It is a 7D vector consisting of the compression ratios of the seven tools (four general-purpose tools and three DNA-specific tools), and a 2D feature that combines the best general-purpose and DNA-specific tools (CRLB). For details of features, see Table 4.4.

The classification performance is shown in Table 7.13. The best performance for predicting Baltimore Classes and ICTV Orders are both achieved using the feature CRA, with error rates of 0.113 and 0.073 respectively.

Baltimore Classes					
Classifier	CRDNA	CRL	CRLL	CRA	CRLB
kNN	0.178 ± 0.007	$\mathbf{0.218 \pm 0.015}$	0.198 ± 0.012	0.128 ± 0.014	0.174 ± 0.009
SVM	$\mathbf{0.162 \pm 0.006}$	0.221 ± 0.014	$\mathbf{0.184 \pm 0.012}$	$\mathbf{0.113 \pm 0.012}$	$\mathbf{0.162 \pm 0.008}$

ICTV Orders					
Classifier	CRDNA	CRL	CRLL	CRA	CRLB
kNN	0.125 ± 0.008	$\mathbf{0.205 \pm 0.014}$	0.132 ± 0.011	0.089 ± 0.014	0.124 ± 0.008
SVM	$\mathbf{0.119 \pm 0.006}$	0.206 ± 0.014	$\mathbf{0.125 \pm 0.012}$	$\mathbf{0.073 \pm 0.011}$	$\mathbf{0.102 \pm 0.008}$

Table 7.13: Classification performance using features based on the compression ratios of DNA specific compression tools.

7.5 Summary of performance

Our results show that classification performance improves notably when extending features from 1-mer to a suitable k-mer. Table 7.14 summarises the best performance achievable by each feature in our study. k-mer counts of ACGT is the best in both Baltimore Class and ICTV Order prediction. MAW is the best in Baltimore

Class prediction (ties with k-mer counts of ACGT) and does well in ICTV Order prediction. k-mer NV and concatenated k-mer NV give equal performance and are very close to k-mer counts. The simplest feature Length is the worst but still gives respectable performance. In all our experiments, ICTV Orders are easier to predict than Baltimore Classes.

Feature	Baltimore Class	ICTV Order
Length	0.204 ± 0.009	0.137 ± 0.013
Counts of SW	0.077 ± 0.008	0.052 ± 0.012
Counts of RY	0.064 ± 0.006	0.030 ± 0.006
Counts of MK	0.090 ± 0.008	0.044 ± 0.008
Concatenated counts of SW, RY and MK	0.038 ± 0.006	0.016 ± 0.006
Counts of ACGT	$\mathbf{0.027 \pm 0.006}$	$\mathbf{0.006 \pm 0.002}$
RTD of ACGT	0.060 ± 0.005	0.014 ± 0.004
Concatenated counts and RTD of ACGT	0.047 ± 0.005	0.009 ± 0.004
k-mer NV of ACGT	0.028 ± 0.005	0.007 ± 0.002
Concatenated k-mer NV of ACGT	0.028 ± 0.005	0.007 ± 0.003
MAW	$\mathbf{0.027 \pm 0.006}$	0.014 ± 0.005
General-purpose compression	0.139 ± 0.006	0.092 ± 0.007
DNA-specific compression	0.113 ± 0.012	0.073 ± 0.011

Table 7.14: The best performance achieved by each feature.

The table entries show the best achievable performance for each feature in the Baltimore Class and ICTV Order experiments, according to Table 7.2 - 7.11, when a suitable k-mer is used. The best performance for each taxonomic scheme is highlighted in bold.

7.6 Identifying difficult viruses

In this section, we conduct leave-one-out experiments to identify viruses whose class labels are difficult to predict. We use 4-mer counts of ACGT as feature and Radial-SVM as classifier since the combination gives the lowest mean error rates in both Baltimore Class and ICTV Order experiments (see Table 7.14). We do not use a majority vote among several different combinations as we did in the previous chapter for two reasons. First, the second best ones (k-mer NV of ACGT and concatenated k-mer NV of ACGT) are practically the same features. Second, the performance of others are much worse, hence a majority vote strategy among several different combinations tends to do more harm than good to the best performing one.

There are 89 sequences in the Baltimore Class experiments and 8 in the ICTV Order experiments that are misclassified by the best feature and classifier combination. Their memberships are summarised in Table 7.15 and 7.16. Among all the 89 viruses identified from the Baltimore Class experiments, 81 are currently unable to be placed into an ICTV Order by the NCBI.

	I	II	III	IV	V	VI	VII
C	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0
M	0	0	0	0	1	0	0
N	0	0	0	5	0	0	0
P	0	0	0	2	0	0	0
T	0	0	0	0	0	0	0
Unlabelled	5	15	15	14	1	16	15

Table 7.15: Membership of viruses whose Baltimore Class labels disagree with the annotated labels.

	C	H	L	M	N	P	T
I	2	2	3	0	0	0	0
II	0	0	0	0	0	0	0
III	0	0	0	0	0	0	0
IV	0	0	0	0	1	0	0
V	0	0	0	0	0	0	0
VI	0	0	0	0	0	0	0
VII	0	0	0	0	0	0	0

Table 7.16: Membership of viruses whose ICTV Order labels disagree with the annotated labels.

7.7 Predicting the Baltimore Class and ICTV Order for currently unlabelled viruses

This section makes predictions of the Baltimore Class and/or ICTV Order for viruses in the dataset currently unlabelled by the schemes, using the best combination of feature and classifier identified in previous studies (4-mer counts of ACGT and Radial-SVM). Table 7.17 summarizes the predicted membership for all currently unlabelled virus sequences in the dataset. Table 7.18 and 7.19 are confusion matrices of classes predicted using the combination of NV12 and MV (the best among all nucleotide-based features) and the combination of 4-mer counts of ACGT and Radial-SVM (the best among all word and compression-based features).

Schemes	**Predicted classes**						
Baltimore Class	I	II	III	IV	V	VI	VII
	36	29	1	8	0	0	0
ICTV Orders	C	H	L	M	N	P	T
	273	80	0	85	12	403	981

Table 7.17: Predicted classes for unlabelled viruses.

The Table shows the number of currently unlabelled viruses predicted in each class.

	I	II	III	IV	V	VI	VII
I	34	0	0	3	0	0	0
II	1	26	0	0	0	0	0
III	0	0	0	1	0	0	0
IV	1	3	1	4	0	0	0
V	0	0	0	0	0	0	0
VI	0	0	0	0	0	0	0
VII	0	0	0	0	0	0	0

Table 7.18: Confusion matrix of predicted Baltimore Classes for unlabelled viruses.

The rows and columns represent predicted classes by NV12 and 4-mer counts respectively. Diagonal entries are the number of predictions made using the two features that agree; off-diagonal entries are the number of predictions that disagree.

	C	H	L	M	N	P	T
C	223	64	0	1	4	9	415
H	6	16	0	0	0	0	0
L	8	0	0	0	1	0	0
M	23	0	0	62	7	80	80
N	6	0	0	5	0	15	1
P	3	0	0	7	0	169	190
T	4	0	0	10	0	130	295

Table 7.19: Confusion matrix of predicted ICTV Orders for unlabelled viruses.

The same as Table 7.18 but for ICTV Order predictions.

7.8 Predicting the virus hosts of non-segmented viruses

Table 7.20 and 7.21 show that both k-mer NV and k-mer counts can give accurate predictions of virus hosts. In general, k-mer counts outperform k-mer NV, which is consistent with our results on Baltimore Class and ICTV Order prediction, emphasising the strength of counts in distinguishing different sequences.

The ability to use the nucleotide composition of virus genome sequences to predict their hosts was previously demonstrated by [147]. The authors correctly identified the kingdom or phylum of the host for $> 95\%$ of picorna-like viruses. The technique also predicted an insect host origin for three novel picorna-like viruses acquired from the stool of an Afghan child. The prediction assisted in the diagnosis of the potential causes of the sickness, which is likely to be the ingestion of insect-contaminated food.

Virus hosts			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	**0.094 ± 0.006**	0.229 ± 0.134	0.098 ± 0.008
2-mer	0.077 ± 0.007	0.146 ± 0.008	**0.061 ± 0.006**
3-mer	0.087 ± 0.007	0.079 ± 0.008	**0.052 ± 0.006**
4-mer	0.088 ± 0.007	0.060 ± 0.008	**0.047 ± 0.006**
5-mer	0.114 ± 0.012	0.052 ± 0.007	**0.045 ± 0.004**
6-mer	0.110 ± 0.009	0.053 ± 0.005	**0.049 ± 0.004**

Table 7.20: Classification error rate of different features and classifiers using the k-mer NV of ACGT.

7.9 Predicting the Baltimore Class of multi-segmented viruses

According to experiments using non-segmented viruses, k-mer counts perform on par with or even better than the k-mer NV. Here we apply the methods to classify a multi-segmented virus into one of the five Baltimore Classes (the multi-segmented

Virus hosts			
Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	**0.100 ± 0.010**	0.183 ± 0.006	0.109 ± 0.008
2-mer	0.068 ± 0.008	0.162 ± 0.008	**0.054 ± 0.008**
3-mer	0.057 ± 0.005	0.078 ± 0.013	**0.039 ± 0.006**
4-mer	0.052 ± 0.007	0.063 ± 0.007	**0.033 ± 0.005**
5-mer	0.051 ± 0.006	0.054 ± 0.005	**0.032 ± 0.004**
6-mer	0.048 ± 0.006	0.053 ± 0.006	**0.042 ± 0.006**

Table 7.21: Classification error rate of different features and classifiers using *k*-mer counts of ACGT.

viruses in the dataset are from five Baltimore Classes and one single ICTV Order, see Table 2.3 for details). To construct features for each virus genome sequence, we count the occurrence of *k*-mers in all the segments. Table 7.22 shows that the methods are well suited to predicting Baltimore Classes.

Feature	**Classifier**		
	k-NN	Linear-SVM	Radial-SVM
1-mer	**0.054 ± 0.015**	0.213 ± 0.036	0.064 ± 0.030
2-mer	**0.023 ± 0.012**	0.105 ± 0.034	0.038 ± 0.013
3-mer	**0.020 ± 0.011**	0.053 ± 0.020	0.031 ± 0.021
4-mer	**0.016 ± 0.014**	0.039 ± 0.013	0.032 ± 0.017
5-mer	**0.016 ± 0.008**	0.037 ± 0.017	0.028 ± 0.015
6-mer	**0.028 ± 0.012**	0.036 ± 0.010	0.035 ± 0.017

Table 7.22: Classification error rate of Baltimore Class prediction for multi-segmented viruses.

7.10 Summary

In this chapter we studied the predictive powers of a wide range of features for genome sequences and classifiers in the task of virus taxonomy.

With features based on *k*-mers, we achieved significant improvements from those based on single nucleotides. We found that richer alphabets improve performance. The performance can be significantly improved using features based on

words (k-mers) with suitable length (k), but performance can deteriorate when an unsuitable k is used. With optimised experimental factors, the best performance for Baltimore Class prediction is achieved by combining 4-mer counts of ACGT with Radial-SVM as well as MAW with k-NN, which tie at an error rate of 0.027 ± 0.006. The best performance for ICTV Order classification is achieved by combining k-mer counts of ACGT ($k = 4$ or 5) with SVM (linear or radial kernel), which tie at an error rate of 0.006 ± 0.002.

Using the best combination of features and classifiers, we identified difficult viruses, provided predictions for currently unlabelled sequences, predicted hosts for non-segmented viruses as well as Baltimore Classes for multi-segmented viruses.

Chapter 8

Hierarchical classification using k-mer counts

In the previous chapters, we have studied the performance of word and compression-based features and supervised learning methods in the non-hierarchical multi-class classification problem of predicting Baltimore Classes, ICTV Orders and hosts of viruses. In this chapter, we extend the task to hierarchical multi-class classification to predict taxonomic classes at all levels of the ICTV hierarchical taxonomic tree.

The current ICTV scheme defines a hierarchy starting at Orders and progressing through Families, Subfamilies, Genera and finally Species. The hierarchical classification task involves classifying reference genome sequences of model species into Orders, Families, Subfamilies and Genera. The task is a hierarchical multi-class classification problem where the class hierarchical structure is a predefined taxonomic tree. The challenges of going deeper are that the number of classes become very large and the number of samples per class becomes very small. In addition, the labelling rate can be very low at certain levels and labels can be missing at intermediate levels.

The main aims of the experiments in this chapter are two-fold. First, we investigate whether incorporating class hierarchical information can improve classification performance. Second, we study whether SSL-based hierarchical classifiers can outperform SL-based counterparts, especially for small classes.

8.1 Methods

8.1.1 Classifiers

Previous chapters showed that the SL method Radial-SVM, together with k-mer counts of ACGT performs the best at classifying viruses into the seven ICTV Orders. Here, we exploit their performance at the deeper levels of the ICTV taxonomic tree by using it as the base classifier in our hierarchical approaches. In addition, we investigate the performance using graph-based SSL as the base classifier and compare it to that of SL.

8.1.2 Modification of the ICTV hierarchical tree

Experiments in this chapter include all the 3,699 non-segmented viruses in the dataset (Table 2.1). Since the labelling rates at certain levels can be very low, e.g. only 0.149 for Subfamily (Table 2.6), we modify the ICTV taxonomic tree with the aim of accommodating the large number of unlabelled viruses. Specifically, we create new classes named "*P_U*", into which viruses belonging to a parent class, called P but with unknown child class are assigned (Fig. 8.1). For example, viruses from Order $Herpesvirales$ but with unknown Family are assigned to a new Family called $Herpesvirales\_U$, and those from $Herpesvirales\_U$ but with unknown Subfamily are assigned to a new Subfamily called $Herpesvirales\_U\_U$. Viruses with unknown Orders are assigned to the class U. Hence, all unlabelled viruses are assigned labels constructed in this way and the viruses from those classes are treated in the same way as the originally labelled ones. Therefore, in the modified tree, all samples are labelled and SL methods can be applied in the usual way.

The modified tree of the 3,699 non-segmented virus species now consists of 8 Orders, 84 Families, 100 Subfamilies and 435 Genera, which we use in our study of hierarchical classification. Table 8.1 and Fig. 8.2 summarise the distributions of class sizes at each level of the original and modified ICTV tree. Class sizes decrease after modification, with the biggest changes occurring at the largest classes at Subfamily and Genus level. Fig. 8.3 displays the number of Species in each taxonomic class ordered by increasing size. The benefits of this modified tree are

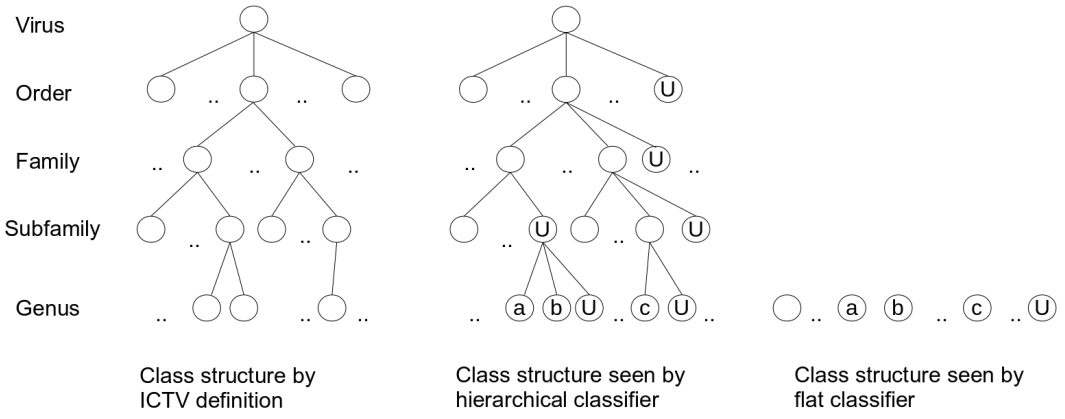


Figure 8.1: Hierarchical structure of virus taxonomic classes.
Class structure by ICTV definition (the original ICTV hierarchical tree), that seen by the hierarchical (modified ICTV hierarchical tree) and flat (modified classes at Genus level) classifiers.

two-fold. First, it incorporates the large number of unlabelled viruses into our study, and second, it divides the unlabelled viruses into smaller and more homogeneous classes. However, the potential disadvantage is that it creates more small classes, leading to imbalanced classes which makes classification difficult.

Level	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Original ICTV tree						
Order	13.00	66.75	141.00	462.40	437.20	1834.00
Family	1.00	3.00	16.00	47.42	53.25	686.00
Subfamily	4.0	14.0	23.5	185.0	50.5	3146.0
Genus	1.00	1.00	3.00	9.94	6.00	1331.00
All levels	1.00	1.00	4.00	30.95	10.00	3146.00
Modified ICTV tree						
Order	13.00	66.75	141.00	462.40	437.20	1834.00
Family	1.00	2.75	13.00	44.04	50.25	686.00
Subfamily	1.00	3.00	12.00	36.99	42.75	686.00
Genus	1.00	1.00	3.00	8.50	6.00	433.00
All levels	1.00	1.00	4.00	23.60	12.50	1834.00

Table 8.1: Summary statistics of the class sizes of the original and modified ICTV tree.

The Table shows the minimum, first quartile, median, mean, third quartile and maximum of the number of model species per class at Order, Family, Subfamily and Genus levels respectively. Rows “All levels” summarise the sizes of the classes in the four levels.

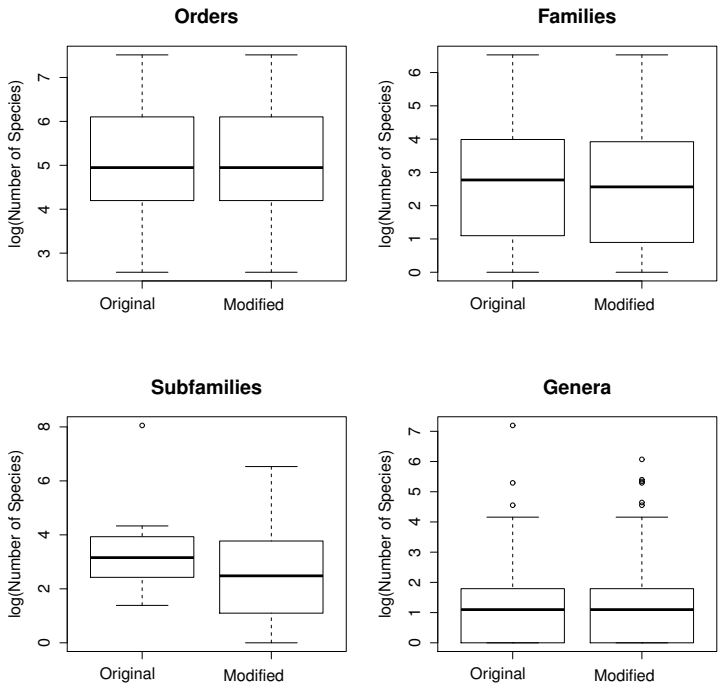


Figure 8.2: Box plot of the class size at each level of the original and modified ICTV tree.

8.1.3 Flat and hierarchical classification

In flat classification, the hierarchical structure still exists but is ignored by the classifier during training and testing, where all classes are treated as coming from a single level. Since the class hierarchy has been predefined, once the class at the deepest level is known, its ancestors can be trivially obtained by definition as part of post-processing. Hence, flat classification involves first classifying a sequence into one of the classes at Genus level and then filling in Order, Family and Subfamily using the ICTV hierarchy.

For hierarchical classification, we use the three local classifiers: LCN, LCL and LCPN (see Section 3.3.3). Class inconsistency can occur during prediction when using the local classifiers LCN and LCL. To correct this inconsistency, we select the most likely path for LCN based on Platt calibration [84], and adopt the sum of votes technique for LCL [1] (for details see Section 3.3.3). We do this because the two

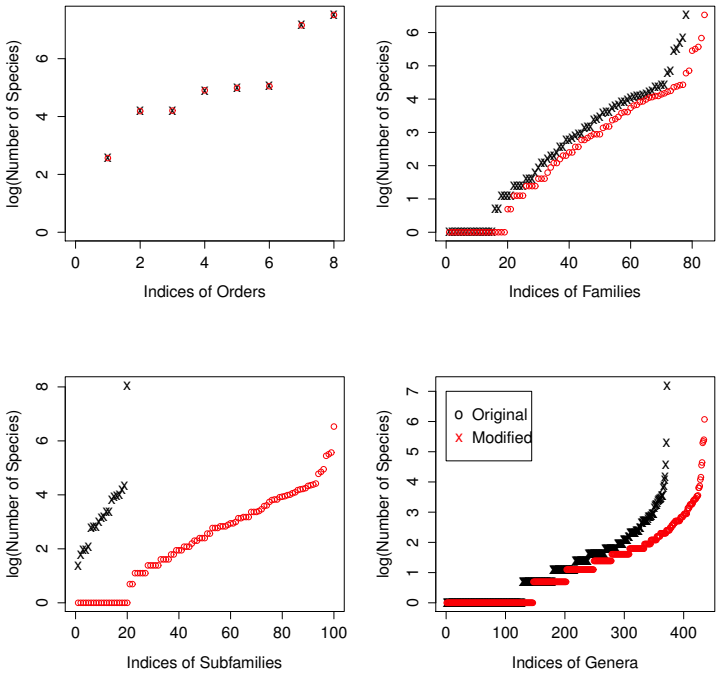


Figure 8.3: The class size at each level of the original and modified ICTV tree.

The x-axis shows the indices of classes ordered by increasing size, and y-axis the log transformed number of Species per class.

methods do not require additional biological information, and can easily be applied to the base classifiers we use here.

8.2 SL-based hierarchical classifiers

8.2.1 Design

In this section, we aim to study the performance of SL-based hierarchical classifiers. Specifically, we construct a hierarchical classifier using an SVM with radial kernel, the best in our previous experiments, to predict the Order, Family, Subfamily and Genus of a model virus species using its genome sequence.

Since classes at deeper levels of the ICTV hierarchical tree are more specific, more sophisticated features may be needed to obtain better performance. Hence, we first study the performance of the different k-mer counts in predicting classes at individual levels of the modified ICTV hierarchical tree, using the same experimen-

tal settings as in Chapter 6. Then, using the best feature for each level, we study whether hierarchical classifiers improve performance from their flat counterparts in predicting the full path of hierarchical classes.

8.2.2 Classification at individual levels of the ICTV scheme

Table 8.2 shows that 4-mer counts consistently outperform the others at every level of the ICTV hierarchical tree, and tie with 5-mer counts at Order and Subfamily. There are two possible reasons why words longer than 4-mer do not improve performance. The first is that longer words may not be more informative than 4-mers for the dataset or the machine learning method, and the second that longer words produce higher dimensional features that increase the chance of overfitting. As a result, 4-mer counts will be used in the following hierarchical experiments. Another observation from the table is that error rates at the Order level are higher than our previous experiments (Table 6.2 and Table 7.4). This is due to the inclusion of unlabelled viruses, which are relatively heterogeneous and occupy 0.496 of the entire dataset, hence increasing the difficulty of the prediction problem. In addition, we can observe that error rates increase significantly at deeper levels, where there are many more classes and fewer viruses per class.

Classifier	Order	Family	Subfamily	Genus
1-mer	0.146 ± 0.005	0.311 ± 0.007	0.330 ± 0.009	0.498 ± 0.009
2-mer	0.082 ± 0.002	0.226 ± 0.020	0.243 ± 0.027	0.418 ± 0.017
3-mer	0.063 ± 0.009	0.191 ± 0.015	0.214 ± 0.025	0.370 ± 0.020
4-mer	$\mathbf{0.048 \pm 0.005}$	$\mathbf{0.149 \pm 0.011}$	$\mathbf{0.173 \pm 0.012}$	$\mathbf{0.331 \pm 0.021}$
5-mer	$\mathbf{0.048 \pm 0.007}$	0.150 ± 0.017	$\mathbf{0.173 \pm 0.019}$	0.394 ± 0.018
6-mer	0.052 ± 0.008	0.184 ± 0.025	0.215 ± 0.030	0.379 ± 0.051

Table 8.2: Classification performance of SVM and k-mer counts at individual ICTV taxonomic levels.

The Table shows the mean and standard deviation of classification error rates at each level of the ICTV hierarchy when different k-mer counts of ACGT are used as features. The best performance in each column is highlighted in bold.

8.2.3 Hierarchical classifiers outperform flat classifiers

Table 8.3 shows classification performance measured without accounting for class hierarchy. The performance of classification at individual levels – that of the 4-mer counts in Table 8.2 – is included as the baseline for comparison.

We observe that all three local classifiers outperform FC at the Order level, but underperform at Genus level, which is contrary to our initial hypothesis that hierarchical classifiers improve performance at every level, especially at deeper levels where there are large numbers of small classes. Experimental results show that the hierarchical classifiers are not as robust as expected, however, the fact that they outperform at high levels despite underperforming at deep levels suggests they are more robust than FC in the sense that they are better able to identify classes similar to the true class, and are less likely to misclassify a virus to a class that is very distant. This is of practical importance when better accuracy at higher levels is important.

The advantages of hierarchical classifiers are emphasised by using hierarchical loss (for details see Section 3.4.2), which measures performance by explicitly accounting for class hierarchy (Table 8.4). Despite the fact that the hierarchical classifier gives slightly higher error rates than FC at Genus level, all three hierarchical losses suggest that hierarchical classifiers outperform FC when accounting for the hierarchy as a whole.

Comparing the three local classifiers, we find that LCN performs the worst, while LCL slightly outperforms LCPN and is the best. At the Family and Subfamily levels, LCL and LCPN outperform FC but LCN underperforms. The main reason LCL outperforms LCN can be found by examining its training procedure for our hierarchical problem, which has a tree structure, single-label per level and leaf prediction (see Fig. 3.1 and Section 3.3.3). The LCN essentially trains a one-vs-all multi-class classifier for each level, whereas LCL trains a one-vs-one multi-class classifier, which has been shown to outperform the one-vs-all scheme for our dataset.

In addition, LCL performs comparably to LCPN. This is mainly because the

inconsistency correction procedure of LCL is similar to the prediction procedure of LCPN. In both methods, a virus is assigned to the class or path having the majority weight, and the total weighted votes of a path tend to be dominated by those from higher level classes. In other words, higher level classes are relatively easier to predict compared to deep level ones, but once a mistake is made at a higher level, it is difficult to effectively correct using post-processing methods without incorporating additional information.

Classifier	Order	Family	Subfamily	Genus
FC	0.058 ± 0.003	0.157 ± 0.013	0.179 ± 0.013	$\mathbf{0.331 \pm 0.021}$
LCN	0.052 ± 0.008	0.169 ± 0.024	0.193 ± 0.021	0.340 ± 0.018
LCL	$\mathbf{0.048 \pm 0.007}$	$\mathbf{0.138 \pm 0.016}$	$\mathbf{0.160 \pm 0.015}$	0.352 ± 0.022
LCPN	$\mathbf{0.048 \pm 0.005}$	0.144 ± 0.010	0.164 ± 0.008	0.332 ± 0.014
Baseline	$\mathbf{0.048 \pm 0.005}$	0.149 ± 0.011	0.173 ± 0.012	$\mathbf{0.331 \pm 0.021}$

Table 8.3: Classification performance of SL-based flat and hierarchical classifiers at individual ICTV taxonomic levels.

The Table shows the mean and standard deviation of classification error rates of the flat classifier (FC), local classifier per node (LCN), local classifier per level (LCL), local classifier per parent node (LCPN) and multi-class classifiers at individual levels (Baseline) using 4-mer counts of ACGT at each ICTV hierarchical level. The best performance in each column is highlighted in bold.

Classifier	l_{Δ}	l_{sibl}	l_{subtr}
FC	0.725 ± 0.047	0.009 ± 0.000	0.015 ± 0.001
LCN	0.753 ± 0.071	0.009 ± 0.001	0.014 ± 0.002
LCL	0.698 ± 0.055	$\mathbf{0.008 \pm 0.001}$	$\mathbf{0.011 \pm 0.002}$
LCPN	$\mathbf{0.688 \pm 0.030}$	$\mathbf{0.008 \pm 0.001}$	0.012 ± 0.002

Table 8.4: Classification performance of SL-based flat and hierarchical classifiers measured using hierarchical loss.

The Table shows the mean and standard deviation of the hierarchical loss of FC, LCN, LCL and LCPN. The best performance in each column is highlighted in bold.

8.3 SSL-based hierarchical classifiers

8.3.1 Design

This section aims to study whether the performance of virus taxonomy can be improved using SSL-based local hierarchical classifiers. SSL has a reputation for good performance on classification problems with only a small number of labelled samples [74].

SSL techniques can be a good solution to the problem of virus taxonomy because they provide a trade-off between prediction accuracy and labelling cost when known samples are sparse in the entire space. This is exactly the situation in the virus kingdom, where known viruses constitute a very small portion of the entire virosphere and their labels can be costly to obtain. In this section, we conduct experiments using SSL-based hierarchical classifiers to predict the class of a virus based on its relationship with others in a network.

As for SL experiments, we use 4-mer counts as features to represent virus genome sequences. In the graph-based SSL experiments, the network is represented as the union graph of the k-nearest neighbour graph (k-NNG) and minimum spanning tree (MST). To predict labels for unlabelled samples in the graph, we use the normalised version of the Laplacian matrix and RBF edge weights with parameter σ = average distance of the nearest neighbours, since they are shown to produce better results in both our previous study and others [148].

We start by demonstrating the advantages of SSL over SL by systematically varying the number of labelled and unlabelled samples. We then construct hierarchical classifiers using SSL as the base classifier and compare their performance with those of SL. Finally, we study which classes benefit the most from an SSL-based method or hierarchical classifier.

8.3.2 SSL outperforms SL when the number of labelled samples are small

We first demonstrate the advantages of SSL over SL using binary-class classification. Here, we show the results for predicting ICTV Orders *Mononegavirales* and

Tymovirales. The two classes are chosen because they have comparable class sizes and are relatively large compared to other Orders, allowing us to vary the number of samples per class in a suitable range. However, similar results can be obtained with other classes.

We randomly draw $m \in \{16, 32, 64, 128\}$ samples from each class from the original dataset, of which $l \in \{2, 4, 8, 16\}$ are then randomly selected as labelled ones. For SL, the l labelled samples in each class are used to train the model and optimise parameters, and the remaining $u = m - l$ samples are used for testing. For SSL, all the m samples in each class are used to construct the graph and labels for the u unlabelled samples are predicted using the methods described in Section 3.2.4. The procedure is repeated 500 times for each pair of m and l to compute the mean and standard deviations of the error rates.

Fig. 8.4 shows that more labelled samples (larger l) lead to better performance for both SL and SSL, whereas SSL consistently outperforms SL for all the values of l in the experiments, more significant for smaller l. The performance of both SSL and SL improve the most as l increases from 2 to 4, while larger values of l give further but less significant improvements. In addition, SSL benefits from unlabelled samples. The performance improves when the number of total samples m increases while keeping l fixed, whereas in contrast, SL performance is unaffected by m.

We then conduct a multi-class classification experiment to predict the seven Baltimore Classes. We randomly split the dataset into training and testing sets in the ratio 75% : 25%. For each split of the dataset, we randomly select $l \in \{2, 4, 8, 16\}$ samples from each class in the training set together with their labels and discard the remaining ones in the set. The total number of samples per class m differs for different classes, which equals l plus the number of samples from that class in the testing set (see Table 2.1). Hence each training set consists of $7l$ labelled samples, which is a small portion of the entire dataset. SL trains the model and tunes parameters using the $7l$ labelled samples, then predicts the labels for the samples in the testing set. SSL constructs a graph using the labelled training samples together with the testing samples, and predicts the unlabelled ones. This procedure is repeated

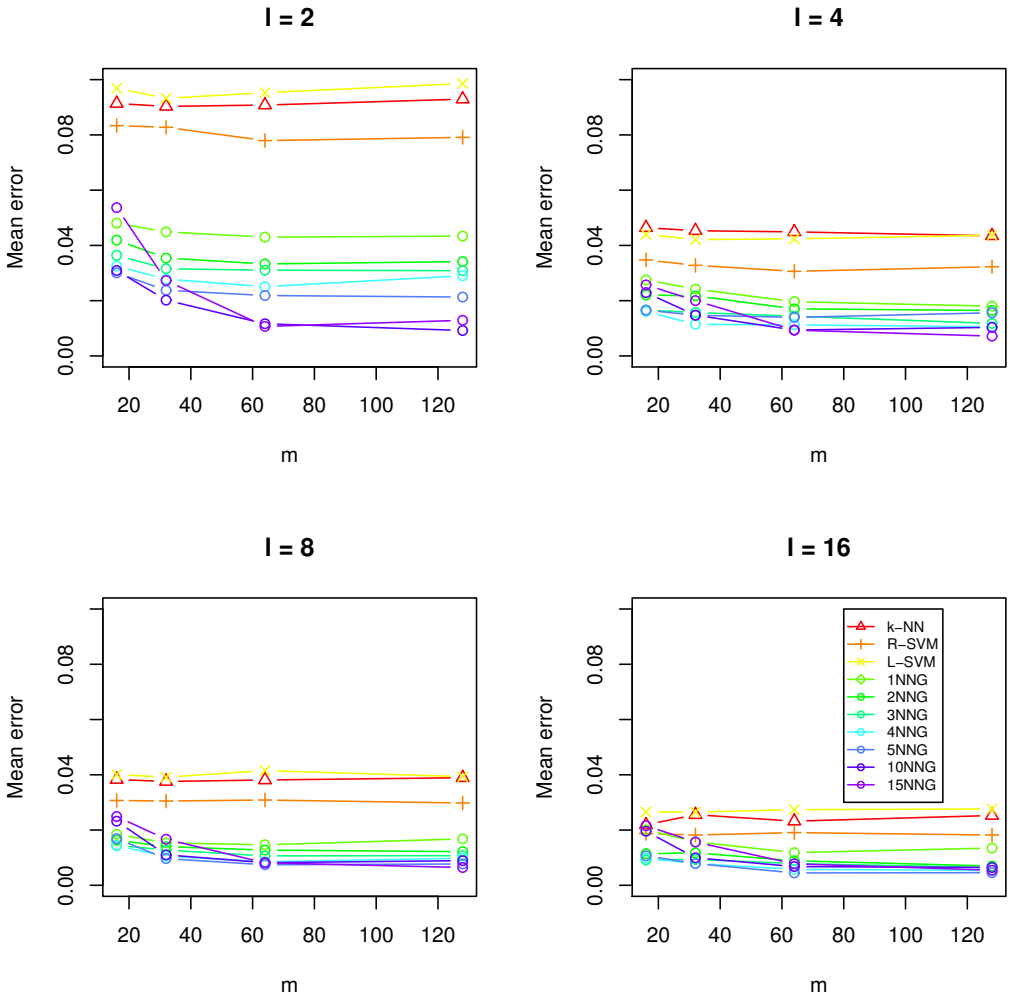


Figure 8.4: Performance of SL and SSL in binary-class classification.

The figure shows the error rates of SL (k-NN, Radial-SVM and Linear-SVM) and SSL (1NNG, ..., 15NNG) in the classification task when the number of total samples per class (m) and labelled samples per class (l) vary.

10 times with random seeds, each giving a different training/testing set division. Finally, we report the mean and standard deviation of the testing error rates over the 10 repeats to summarise the classification performance on the overall dataset. Our earlier results show that including all the unlabelled samples in the training set into the SSL graph can improve performance. However, inverting the matrix of a large graph is computationally expensive, hence we exclude them in the following experiments to reduce computational cost.

Table 8.5 shows that SSL outperforms SL in most cases, more significantly for smaller l. As shown in Fig. 8.4, performance improves the most when l increases from 2 to 4. When l = 16, SSL still outperforms SL but the advantage becomes less prominent.

Classifier	$l = 2$	$l = 4$	$l = 8$	$l = 16$
SL				
k-NN	0.532 ± 0.085	0.405 ± 0.071	0.304 ± 0.044	0.243 ± 0.026
L-SVM	0.532 ± 0.112	0.422 ± 0.103	0.353 ± 0.080	0.293 ± 0.054
R-SVM	0.437 ± 0.124	0.292 ± 0.057	0.251 ± 0.031	0.212 ± 0.022
SSL				
1NNG	0.474 ± 0.155	0.341 ± 0.086	0.262 ± 0.083	$\mathbf{0.193 \pm 0.037}$
2NNG	$\mathbf{0.415 \pm 0.165}$	$\mathbf{0.287 \pm 0.058}$	$\mathbf{0.239 \pm 0.033}$	$\mathbf{0.181 \pm 0.028}$
3NNG	$\mathbf{0.394 \pm 0.129}$	$\mathbf{0.277 \pm 0.044}$	$\mathbf{0.241 \pm 0.039}$	$\mathbf{0.183 \pm 0.028}$
4NNG	$\mathbf{0.394 \pm 0.139}$	0.297 ± 0.035	$\mathbf{0.225 \pm 0.025}$	$\mathbf{0.189 \pm 0.032}$
5NNG	$\mathbf{0.373 \pm 0.117}$	$\mathbf{0.280 \pm 0.059}$	$\mathbf{0.236 \pm 0.030}$	$\mathbf{0.189 \pm 0.026}$
10NNG	$\mathbf{0.410 \pm 0.137}$	$\mathbf{0.266 \pm 0.040}$	$\mathbf{0.238 \pm 0.022}$	$\mathbf{0.201 \pm 0.022}$
15NNG	$\mathbf{0.377 \pm 0.101}$	$\mathbf{0.265 \pm 0.045}$	0.255 ± 0.025	0.220 ± 0.023
20NNG	$\mathbf{0.401 \pm 0.150}$	0.297 ± 0.060	0.269 ± 0.039	0.225 ± 0.022

Table 8.5: Performance of SL and SSL in multi-class classification.

The table shows the mean and standard deviation based on 10 random splits of training and testing sets. Cases where SSL outperforms SL are highlighted in bold.

8.3.3 SSL outperforms SL in hierarchical classification

Our hypothesis is that hierarchical local classifiers using graph-based SSL as the base classifier outperform those using SL for problems with a small number of labelled samples in a large number of classes. In the experiments, m for a class equals the number of samples in that class in the dataset (Table 2.1) and l equals the number of samples of that class in the training set. The value of m and l tend to be different for each class.

Comparing Table 8.6 vs Table 8.3, and Table 8.7 vs Table 8.4, we can observe that SSL-based approaches give lower errors in most cases. When performance is measured at individual levels (Table 8.6 vs Table 8.3), the improvement from SSL-based classifiers is more significant at Order level than deeper levels, where they occasionally underperform compared to the SL-based classifiers, for instance LCL

and LCPN at Family, and LCN at the Genus levels. The SSL-based hierarchical classifier does not always reduce the misclassification rates at deeper levels, however, it reduces the chance of misclassifying a virus to a distant class, by consistently giving lower error rates at higher levels. The advantages become more noticeable if class hierarchy is accounted for when measuring misclassification (Table 8.7 vs Table 8.4).

Classifier	Order	Family	Subfamily	Genus
FC	0.055 ± 0.005	0.154 ± 0.009	0.169 ± 0.012	0.335 ± 0.013
LCN	0.050 ± 0.003	0.150 ± 0.008	0.168 ± 0.010	0.339 ± 0.011
LCL	$\mathbf{0.044 \pm 0.004}$	$\mathbf{0.145 \pm 0.005}$	$\mathbf{0.159 \pm 0.005}$	$\mathbf{0.328 \pm 0.018}$
LCPN	0.047 ± 0.002	$\mathbf{0.145 \pm 0.003}$	0.160 ± 0.005	0.331 ± 0.010
Baseline	0.047 ± 0.002	0.153 ± 0.008	0.169 ± 0.010	0.335 ± 0.013

Table 8.6: Classification performance of SSL-based flat and hierarchical classifiers at individual ICTV taxonomic levels.

The same as Table 8.3 but using graph-based SSL as the base classifier. The best performance in each column is highlighted in bold.

Classifier	l_{Δ}	l_{sibl}	l_{subtr}
FC	0.713 ± 0.038	0.009 ± 0.001	0.014 ± 0.001
LCN	0.706 ± 0.031	$\mathbf{0.008 \pm 0.000}$	$\mathbf{0.012 \pm 0.001}$
LCL	$\mathbf{0.675 \pm 0.027}$	$\mathbf{0.008 \pm 0.000}$	$\mathbf{0.012 \pm 0.001}$
LCPN	0.682 ± 0.018	$\mathbf{0.008 \pm 0.000}$	0.013 ± 0.000

Table 8.7: Classification performance of SSL-based flat and hierarchical classifiers measured using hierarchical loss.

The same as Table 8.4 but using SSL as the base classifier.

8.3.4 SSL outperforms SL for small classes in hierarchical classification

We now study which classes benefit from the use of a hierarchical classifier or SSL the most. We first divide the 627 taxonomic classes at all four ICTV levels in the dataset into four disjoint groups according to their class size such that each group has roughly the same number of classes. Group 1 contains classes with no more than 3 samples, group 2 contains classes with 4-6 samples, group 3 contains classes

with 7-20 samples, and group 4 contains classes with at least 21 samples. The number of classes in each group are 300, 104, 114, and 109 respectively. Table 8.8 summarises the number of classes and samples in each group.

Samples per class	1 - 3	4 - 6	7 - 20	≥ 21
Number of Orders	0	0	1	7
Number of samples	0	0	13	3,686
Number of Families	25	8	17	34
Number of samples	35	37	224	3,403
Number of Subfamilies	27	11	24	38
Number of samples	39	52	308	3,300
Number of Genera	248	85	72	30
Number of samples	396	420	829	2,054
Total number of classes	300	104	114	109
(Total number of samples	470	509	1,374	12,443

Table 8.8: Number of classes and samples in each group.

We then compute the overall error rates for classes in each group for the FC and hierarchical classifiers. According to the results shown in Table 8.9, for both SL and SSL, there is no clear evidence of improvement over FC when using hierarchical classifiers for classes in each group since the standard deviations of error rates are large compared to the means.

When comparing SSL and SL based methods, the most significant improvements occur for the group of classes with 4-6 samples. Smaller classes give consistently higher error rates that are unlikely to be reduced by any approach. Larger classes, in contrast, give lower error rates with all classifiers and the advantages of hierarchical classifiers or SSL is unnoticeable. The reason SSL does not improve performance for classes with fewer than 4 samples ($m < 4$), as opposed to the results in Fig. 8.4 and Table 8.5, is that samples in those small classes may not occur at all in the training set in this experiment, whereas previous experiments ensure l training samples for each class.

However, the total number of samples in the group of classes with 4-6 samples constitute only a small proportion of the entire dataset (Table 8.8), and the overall

performance of a classifier is dominated by the results from large classes.

Based on the above studies, we conclude that hierarchical classifiers and SSL are able to improve performance, but their main advantage is probably their capability to assign viruses into classes that share the same ancestors as the correct classes in cases of misclassification.

Samples per class	1 - 3	4 - 6	7 - 20	≥ 21
SVM				
FC	0.281 ± 0.487	0.226 ± 0.392	0.099 ± 0.172	0.037 ± 0.064
LCN	$\mathbf{0.273 \pm 0.473}$	0.223 ± 0.387	0.153 ± 0.266	0.039 ± 0.068
LCL	0.303 ± 0.525	0.209 ± 0.361	0.101 ± 0.176	$\mathbf{0.034 \pm 0.059}$
LCPN	0.298 ± 0.516	0.186 ± 0.321	$\mathbf{0.092 \pm 0.159}$	0.038 ± 0.065
SSL				
FC	0.298 ± 0.516	$\mathbf{0.157 \pm 0.271}$	0.099 ± 0.172	0.038 ± 0.066
LCN	0.295 ± 0.511	0.174 ± 0.301	0.107 ± 0.185	0.037 ± 0.064
LCL	0.295 ± 0.511	0.165 ± 0.286	0.107 ± 0.185	0.036 ± 0.062
LCPN	0.298 ± 0.516	0.162 ± 0.281	0.108 ± 0.187	0.036 ± 0.062

Table 8.9: Classification error rates for classes in each group.

The table shows error rates and standard deviations for classes of a given size range. Results with the lowest error rates in each column are highlighted in bold.

8.4 Summary

In this chapter, we extended previous studies to predict deeper levels of ICTV hierarchical taxonomic classes. It is evident that incorporating hierarchical information can improve performance for SL-based hierarchical classifiers, and their advantages are emphasised by hierarchical measures, which merit the importance of accurate predictions at higher levels. However, when measuring performance at individual levels separately, we found that hierarchical classifiers consistently outperform at Order level but can underperform at Genus level. This suggests that in cases of misclassification, hierarchical classifiers are less likely to assign viruses to a class very distant from the true class compared to non-hierarchical approaches.

Similar behaviour can be observed from SSL-based hierarchical classifiers. We also found that SSL-based approaches can outperform SL-based ones, especially for classes with 4-6 sequences. The improvement to the overall performance is small

because the number of viruses from these small classes constitute only a small proportion of the entire dataset. However, SSL is better able to assign viruses into classes that share the same ancestors as the correct classes in cases of misclassification at descendant levels.

Chapter 9

Conclusions

In this chapter, we will summarise our work and contributions, and discuss current and future work.

9.1 Summary

This thesis contributes to the application of machine learning techniques to genome sequence-based virus taxonomy. Our study uses the virus reference sequence (RefSeq) dataset provided by NCBI, a dataset widely used by virologists.

We started our study by assessing the extent to which information provided by genome sequences can be used to distinguish viruses from different taxonomic classes. We conducted a thorough analysis of the virus genome sequence dataset, exceeding previous studies in scale and depth. We analysed the statistical properties of the nucleotide composition of genome sequences for all currently discovered virus species, and studied the predictive power of different feature representations for genomes based on their statistical properties. We also provided visualisations of all the sequences in the dataset in two-dimensional space. We have shown that the distributions of global composition- and location-related statistics of nucleotides and words, as well as MAW and compression ratios of the sequences in the dataset appear distinguishable between different classes, for ICTV Orders in particular, and Baltimore Classes to a lesser extent. A single-variable feature Length can already distinguish between different classes and plays an important role in differentiating them.

Next, we investigated the predictive powers of different feature representations of genome sequences and classifiers in the task of virus taxonomy. We are the first to conduct a systematic study to compare and contrast the predictive powers of various combinations of features and classifiers, from simple nucleotide statistics to sophisticated features based on *k*-mers and compressibility. Length already gives respectable performance, while more sophisticated features of nucleotides and *k*-mers improve performance, especially for small classes. With optimised experimental factors, the best performance for ICTV Order prediction is achieved by a combination of 4-mer or 5-mer counts and SVM, and the best for Baltimore Class prediction is achieved using 4-mer count and SVM as well as MAW and k-NN, both giving significant improvements to the current state of the art. Using the best experimental settings identified in the study, we predicted labels for currently unlabelled sequences. We also extended our study to predict virus host and taxonomic classes for multi-segmented viruses.

Finally, we extended our prediction to deeper levels of the ICTV hierarchical taxonomic tree. we are the first to explicitly incorporate hierarchical information and apply SSL-based hierarchical classifiers to the dataset for predicting the ICTV classes. It is evident that classifiers incorporating class hierarchical information improve performance when assessed using hierarchical measures. When measuring performance at individual levels, the former consistently outperform at higher levels despite the potential for slightly worse performance at deeper levels. We showed that SSL-based classifiers consistently outperform SL-based ones and that the largest improvements are likely to come from small classes with 4-6 viruses.

9.2 Discussion

There are several aspects of the research we would like to highlight here.

9.2.1 Missing labels in the taxonomic tree

In our study of hierarchical classification, we proposed an approach to accommodate the large number of unlabelled sequences in the ICTV hierarchical taxonomy by assigning new labels to unlabelled ones by appending “U” to their parent’s la-

bels. This approach essentially creates new classes for unlabelled sequences to fill in missing labels, hence regular SL methods can be applied. The graph-based SSL does not assist in filling in the missing labels, but only in label prediction. An alternative is to explore hierarchical SSL methods that assist in both the missing label and label prediction problems, details of which will be discussed in the next section.

9.2.2 Imbalanced classes

The virus genome sequence dataset contains imbalanced classes, with larger classes having significantly more samples than smaller ones. Small classes consistently give higher error rates, which is a general issue associated with almost any classifier, as their aim is to minimise the overall error rates. From a statistical perspective, small classes suffer from insufficient information of the distribution of their samples and the small number of samples may not be representative of their population, hence models built on them are less able to generalise to unseen samples.

Efforts were made to improve performance for small classes. Initially, we tried resampling methods [149, 150] but found that while they can be beneficial to small classes, they are harmful for large ones as well as the overall performance. However, we managed to apply a number of strategies that can effectively improve performance for not only small classes but also overall. Firstly, we find that using a majority voting scheme among a number of competing classification techniques can effectively reduce error rates for difficult viruses, which are typically from small classes or class boundaries. Secondly, more sophisticated features and classifiers are useful especially for small classes. We observe noticeable improvements from Length to NV12, then to k-mer features. Third, including the hierarchical relationship improves performance since more information is provided for small classes. For example, viruses are less likely to be misclassified into a distant class when using hierarchical classifiers. Finally, exploiting the relationship between other classes can also contribute to better performance. We showed that graph-based SSL outperforms SL for small classes, especially those with 4-6 samples per class.

However, insufficient data is an intrinsic disadvantage of small classes that is unlikely to be eliminated completely. The same problem also holds for human

experts, who are more likely to misclassify unfamiliar or rare viruses. A practical implication of applying classifiers trained using a biased data set is that predictions of a small class label imply a high level of confidence of its membership, whereas predictions of a large class label do not imply so. As a result, for the currently unlabelled viruses, we would be more confident in the predicted class membership of viruses if they are classified into a small class such as Baltimore Class III and ICTV Order N, H and L.

9.2.3 Validation using the latest dataset

The NCBI RefSeq dataset is continuously updated as new genomes are sequenced and scientific discoveries are made. A number of changes were made to the dataset since we started the work in this thesis. A recent version downloaded on 20^{th} August 2017 contains 3,721 non-satellite non-segmented viruses, an increase of 22 compared to the dataset used in the thesis. In the later version of the dataset, 101 new viruses are added and 79 in the earlier version are removed.

Using the later version, we validate our predictions for difficult viruses and previously unlabelled ones. Recall that in Chapter 6, we use NV12 as the feature and MV as the classifier to predict the Baltimore Class and ICTV Order of a virus. Among the 206 misclassified viruses in the Baltimore Class prediction experiments, 2 are removed, 5 become unlabelled, and 9 are assigned different labels, of which 2 agree with the predictions. Among the 61 misclassified viruses in the ICTV Order prediction experiments, 1 is removed. In similar studies using 4-mer counts of ACGT and SVM in Chapter 7, among the 89 misclassified viruses in the Baltimore Class prediction experiments, 1 is removed, 3 become unlabelled, and 6 are assigned different labels, of which 2 agree with the predictions. There are no changes for the 8 misclassified viruses in the ICTV Order prediction experiments.

When validating the predictions for unlabelled viruses, we find that 10 out of 74 viruses with unknown Baltimore Classes in the earlier version of the dataset become assigned (9 from Class I and 1 from Class II), all of which are correctly predicted using both the combination of NV12 and MV and the combination of 4-mer count and SVM; 17 out of 1,834 viruses with unknown ICTV Orders are assigned

(16 from *Caudovirales* and 1 from *Tymovirales*), of which 16 are correctly predicted and 1 is misclassified. The misclassified virus ($Xaxnthomonasphage Cf1c$) has significant changes in its labels, being assigned to a completely different Baltimore Class, ICTV Family and Genus in the later version.

9.3 Future work

There are a number of research topics that follow on naturally from the work in this thesis.

9.3.1 Features of sequences

A number of future research directions are related to features of sequences. In Chapter 7 and 8 of the thesis, we studied features based on the statistics of k-mers, obtained by exactly matching short sequences of k consecutive nucleotides with the given k-mers exactly. A natural extension is to allow mismatches with gapped k-mers [151, 152, 50, 153], which are short sequences similar to the given k-mers but containing different patterns or having different length.

In addition, the MAW methods studied in Chapter 7 are based on short nucleotide sequences that are possible but absent in the genome sequence. A complementary approach is to explore the predictive powers of patterns that are present, such as patterns satisfying the criteria proposed in [154, 155]: frequent and distinctive in at least one class but not redundant. Being distinctive means a feature should be significantly correlated with at least one class and no-redundancy refers to the specificity and generalisability of the patterns.

In our current work, the feature variables were all manually designed. In the future, we can explore the efficacy of feature representations generated by automated methods. For instance, the word2vec methodology [156], originally proposed for natural language processing, generates vector representations for words using neural net language models. A similar model has been successfully applied to construct features for biological sequences [157].

In our k-mer based features, the maximum k is six, which is a relatively small number. Our results suggest that features based on 4-mers or 5-mers consistently

outperform those based on 6-mers. However, [158] has found that $k = 9$ is optimal in terms of information content. Therefore, using $k > 6$ together with effective feature selection methods can potentially improve performance by extracting information contained in longer words that is relevant to classification. When a large k is used, feature selection or reduction becomes important. First, direct classification using the complete features can be inefficient or even impossible. Second, selecting the small number of features contributing most to classification improves the interpretability of results and is of biological importance. A summary of k-mer based feature selection methods for sequence classification can be found in [159]. Another problem associated with large k-mer features is the computational resources required for constructing them. Similar works [116, 152, 153] achieve efficient computation for the collection of k-mers by building a suffix tree [160]. We can explore similar methods to collect not only k-mers but also positional and other statistical information.

9.3.2 SSL

SSL is another direction for future work. In our graph-based SSL methods, we construct graphs using the union graph of the kNNG and MST based on our practical experience and recommendations from [76]. Since the quality of the graph is important to the performance of SSL, a graph that better reflects the relationship between sequences can reduce prediction error rates. One approach is to apply clustering methods to form a natural grouping of sequences and another is to explore more suitable distance measures [11, 161].

Our study of graph-based SSL hierarchical classification only focuses on local classifiers, which work by combining multiple non-hierarchical classifiers. An alternative is to construct a global classifier that outputs predictions for all classes from a single optimisation. To achieve this, the current graph-based SSL needs to be modified to address multi-class problems with hierarchical class structures. One way to do this is to explore class-specific feature variables as described in [162, 163]. Since global hierarchical classifiers involve a single large optimisation problem, direct matrix inversion in graph-based SSL can become computationally

intractable, and hence efficient computation on graphs becomes necessary [164].

In this thesis, we only studied one type of graph-based SSL method, however others can be explored (see [76]). Various SSL-based hierarchical classifiers have been applied to protein function predictions [165], but not in the virus genome sequence dataset or the features we use. The most widely used method is self-training. It first trains an SL model using the labelled samples, then uses the trained model to predict the unlabelled ones in the training set. Hence, the unlabelled samples in the training set are assigned putative labels. Next, it trains a final model using all samples in the training set together with their labels. It has been applied to different biological sequences, e.g. protein function prediction [166], gene function prediction [167] and virus classification [52]. As our results suggest that SVM is the best SL approach for our problem, the S3VM [168], a semi-supervised version of SVM based on self-training can be a suitable approach. However, one disadvantage of self-training is the assumption that all unlabelled samples are members of the existing classes, which is not true in our virus taxonomy problem [5]. Another is that it requires more than l samples in the training set to improve on the standard SL approach, whereas graph-based SSL can already improve with only l labelled samples. A potentially better approach may be semi-supervised clustering, e.g. SSL-EM [169], which first groups labelled and unlabelled training samples into different clusters and then predicts the membership of the testing samples.

Bibliography

[1] Bruno C Paes, Alexandre Plastino, and Alex A Freitas. Improving local per level hierarchical classification. *Journal of Information and Data Management*, 3(3):394–409, 2012.

[2] A. M. Q. King, M. J. Adams, E. B. Carstens, and E. J. Lefkowitz, editors. *Virus taxonomy: classification and nomenclature of viruses: Ninth Report of the International Committee on Taxonomy of Viruses*. Elsevier/Academic Press, London, 2012.

[3] M. Krupovič and D. H. Bamford. Order to the viral universe. *Journal of Virology*, 84(24):12476–12479, 2010.

[4] P. Simmonds. Methods for virus classification and the challenge of incorporating metagenomic sequence data. *Journal of General Virology*, 96(6):1193–1206, 2015.

[5] D. M. Knipe and P. M. Howley. *Fields virology*. Lippincott Williams and Wilkins, Philadelphia, PA, 2013.

[6] E. Kejnovsky and E. N. Trifonov. Horizontal transfer - imperative mission of acellular life forms, acytota. *Mobile Genetic Elements*, 6(2), 2016.

[7] E. N. Trifonov and E. Kejnovsky. Acytota - associated kingdom of neglected life. *Journal of Biomolecular Structure and Dynamics*, 34(8):1641–1648, 2016.

[8] T. Hyypi, T. Hovi, N. J. Knowles, and G. Stanway. Classification of enteroviruses based on molecular and biological properties. *Journal of General Virology*, 78(1):1–11, 1997.

[9] J. L. Geoghegan, A. M. Senior, F. Di Giallonardo, and E. C. Holmes. Virological factors that increase the transmissibility of emerging human viruses. *Proceedings of the National Academy of Sciences*, 113(15):4170–4175, 2016.

[10] D. Baltimore. Expression of animal virus genomes. *Bacteriol Review*, 35(3):235–241, 1971.

[11] S. Vinga and J. Almeida. Alignment-free sequence comparison – a review. *Bioinformatics*, 19(4):513–523, 2003.

[12] I. Schwende and T. D. Pham. Pattern recognition and probabilistic measures in alignment-free sequence analysis. *Briefings in Bioinformatics*, 15(3):354–368, 2014.

[13] K. Song, J. Ren, G. Reinert, M. Deng, M. S. Waterman, and F. Sun. New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Briefings in Bioinformatics*, 15(3):343–353, 2014.

[14] Peter Simmonds, Mike J Adams, Mária Benkő, Mya Breitbart, J Rodney Brister, Eric B Carstens, Andrew J Davison, Eric Delwart, Alexander E Gorbalenya, Balázs Harrach, et al. Consensus statement: Virus taxonomy in the age of metagenomics. *Nature Reviews Microbiology*, 15:161168, 2017.

[15] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambridge University Press, 1998.

[16] M. S. Waterman. *Introduction to Computational Biology: Maps, Sequences, and Genomes: Interdisciplinary Statistics*. Chapman and Hall/CRC, Boca Raton, FL, 1995.

[17] B. Muhire, D. P. Martin, J. K. Brown, J. Navas-Castillo, E. Moriones, F. M. Zerbini, R. Rivera-Bustamante, V. G. Malathi, R. W. Briddon, and A. Varsani. A genome-wide pairwise-identity-based proposal for the classification of viruses in the genus Mastrevirus (family Geminiviridae). *Archives of Virology*, 158(6):1411–1424, 2013.

[18] Y. Bao, V. Chetvernin, and T. Tatusova. Improvements to pairwise sequence comparison (PASC): a genome-based web tool for virus classification. *Archives of Virology*, 159(12):3293–3304, 2014.

[19] B. M. Muhire, A. Varsani, and D. P. Martin. SDT: a virus classification tool based on pair-wise sequence alignment and identity calculation. *PLoS One*, 9(9), 2014.

[20] K. Katoh, K. Misawa, K. Kuma, and T. Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, 30(14):3059–3066, 2002.

[21] R. C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004.

[22] M. A. Larkin, G. Blackshields, N. P. Brown, R. Chenna, P. A. McGettigan, H. McWilliam, F. Valentin, I. M. Wallace, A. Wilm, R. Lopez, J. D. Thompson, T. J. Gibson, and D. G. Higgins. Clustal w and clustal x version 2.0. *Bioinformatics*, 23(21):2947–2948, 2007.

[23] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[24] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[25] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[26] Eugene W Myers and Webb Miller. Optimal alignments in linear space. *Bioinformatics*, 4(1):11–17, 1988.

[27] M Van Regenmortel, D Bishop, C Fauquet, M Mayo, and J Maniloff. *Virus classification by pairwise sequence comparison (PASC)*. Academic Press, San Diego, 2010.

[28] M. O. Dayhoff, R. Schwartz, and B. Orcutt. *A model of evolutionary change in proteins*, volume 5, pages 345–352. National Biomedical Research Foundation, 1978.

[29] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[30] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.

[31] W. Just. Computational complexity of multiple sequence alignment with sp-score. *Journal of computational biology*, 8(6):615–623, 2001.

[32] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg. Alignment of whole genomes. *Nucleic Acids Research*, 27(11):2369–2376, 1999.

[33] Yadvir Kaur and Neelofar Sohi. Comparison of different sequence alignment methods-a survey. *International Journal of Advanced Research in Computer Science*, 8(5):2307–2311, 2017.

[34] Michael Lynch. Intron evolution as a population-genetic process. *Proceedings of the National Academy of Sciences*, 99(9):6118–6123, 2002.

[35] Ying-Xin Zhang, Kim Perry, Victor A Vinci, Keith Powell, Willem PC Stemmer, and Stephen B del Cardayré. Genome shuffling leads to rapid phenotypic improvement in bacteria. *Nature*, 415(6872):644–646, 2002.

[36] Teresa K Attwood. The babel of bioinformatics. *Science*, 290(5491):471–473, 2000.

[37] B. E. Blaisdell. A measure of the similarity of sets of sequences not requiring sequence alignment. *Proceedings of the National Academy of Sciences*, 83(14):5155–5159, 1986.

[38] C. Yu, Q. Liang, C. Yin, R. L. He, and S. S.-T. Yau. A novel construction of genome space with biological geometry. *DNA Research*, 17(3):155–168, 2010.

[39] M. Deng, C. Yu, Q. Liang, R. L. He, and S. S.-T. Yau. A novel method of characterizing genetic sequences: genome space with biological distance and applications. *PLoS ONE*, 6(3), 2011.

[40] J. Wen, R. H. F. Chan, S.-C. Yau, R. L. He, and S. S. T. Yau. K-mer natural vector and its application to the phylogenetic analysis of genetic sequences. *Gene*, 546(1):25–34, 2014.

[41] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(2):149–154, 2001.

[42] G. Hampikian and T. Andersen. Absent sequences: Nullomers and primes. In *Pacific Symposium on Biocomputing*, pages 355–366, 2007.

[43] A. J. Pinho, P. J. Ferreira, S. P. Garcia, and J. M. Rodrigues. On finding minimal absent words. *BMC Bioinformatics*, 10(137), 2009.

[44] P. He and J. Wang. Numerical characterization of DNA primary sequence. *Internet Electronic Journal of Molecular Design*, 1(12):668–674, 2002.

[45] C. Yin and S. S. Yau. An improved model for whole genome phylogenetic analysis by Fourier transform. *Journal of Theoretical Biology*, 382:99–110, October 2015.

[46] R. Zhang and C. T. Zhang. A brief review: the Z-curve theory and its application in genome analysis. *Current Genomics*, 15(2):78–94, 2014.

[47] J. S. Almeida. Sequence analysis by iterated maps, a review. *Briefings in Bioinformatics*, 15(3):369–375, 2014.

[48] T. Hernandez and J. Yang. Descriptive statistics of the genome: phylogenetic classification of viruses. Archive of Cornell University Library, Sep 2013.

[49] Rong She, Fei Chen, Ke Wang, Martin Ester, Jennifer L Gardy, and Fiona SL Brinkman. Frequent-subsequence-based prediction of outer membrane proteins. In *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining*, pages 436–445. ACM, 2003.

[50] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

[51] Konstantinos Blekas, Dimitrios I Fotiadis, and Aristidis Likas. Motif-based protein sequence classification using neural networks. *Journal of Computational Biology*, 12(1):64–82, 2005.

[52] C. Yu, T. Hernandez, H. Zheng, S.-C. Yau, H.-H. Huang, R. L. He, J. Yang, and S. S.-T. Yau. Real time classification of viruses in 12 dimensions. *PLoS ONE*, 8(5), 2013.

[53] I. Borozan, S. Watt, and V. Ferretti. Integrating alignment-based and alignment-free sequence similarity measures for biological sequence classification. *Bioinformatics*, 31(9):1396–1404, 2015.

[54] K. Pruitt, G. Brown, T. Tatusova, and D. Maglott. Chapter 18: The Reference Sequence (RefSeq) Database. 2012.

[55] D. A. Benson, M. Cavanaugh, K. Clark, I. Karsch-Mizrachi, D. J. Lipman, and E. W. Ostell, J.and Sayers. Genbank. *Nucleic Acids Research*, 41(Database issue):D36–D42, 2013.

[56] NCBI ftp site, September 2015. Retrieved September 18, 2015, from ftp://ftp.ncbi.nlm.nih.gov/genomes/Viruses/.

[57] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, 2006.

[58] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, Cambridge, 2006.

[59] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, New York, NY, 2009.

[60] M. W. Libbrecht and W. S. Noble. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16:321–332, 2015.

[61] Z. Xing, J. Pei, and E. Keogh. A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter*, 12(1):40–48, November 2010.

[62] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[63] class: Functions for classification, Augest 2015. Retrieved Augest 11, 2015, from https://cran.r-project.org/web/packages/class/index.html.

[64] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[65] randomforest: Breiman and cutler's random forests for classification and regression, October 2015. Retrieved October 6, 2015, from https://cran.r-project.org/web/packages/randomForest/index.html.

[66] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Wadsworth and Brooks/Cole Advanced Books and Software, Monterey, CA, 1984.

[67] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[68] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, 2000.

[69] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, 2004.

[70] V. N. Vapnik. *Statistical learning theory*. Wiley-Interscience, New York, NY, September 1998.

[71] J. P. Vert, K. Tsuda, and B. Schölkopf. *A primer on kernel methods*, pages 35–70. MIT Press, Cambridge, MA, 2004.

[72] C. C. Chang and C. J. Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:3:27, 2011.

[73] G. Fung and O. L. Mangasarian. A feature selection newton method for support vector machine classification. *Computational Optimization and Applications Journal*, 28(2):185–202, 2004.

[74] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912–919, 2003.

[75] C. Chapelle, A. Zien, and B. Schölkopf. *Semi-supervised learning*. MIT Press, Cambridge, MA, 2006.

[76] X. Zhu and A. B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan and Claypool, San Rafael, CA, 2009.

[77] The r project for statistical computing, July 2015. Retrieved July 20, 2015, from https://www.r-project.org.

[78] e1071: Misc functions of the department of statistics, probability theory group (formerly: E1071), tu wien, Augest 2015. Retrieved Augest 27, 2015, from https://cran.r-project.org/web/packages/e1071/index.html.

[79] penalizedsvm: Feature selection svm using penalty functions, October 2012. Retrieved May 18, 2016, from https://cran.r-project.org/web/packages/penalizedSVM/index.html.

[80] Genome sequence-based virus taxonomy using machine learning, October 2017. Retrieved October 1, 2017, from https://github.com/TingtingWang2011/VirusTaxonomy.

[81] C. N. Silla and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1):31–72, 2011.

[82] A. A. Freitas and A. Carvalho. A tutorial on hierarchical classification with applications in bioinformatics. In D. Taniar, editor, *Research and Trends in Data Mining Technologies and Applications*, pages 175–208, 2007.

[83] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Hierarchical classification: Combining bayes with svm. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 177–184, New York, NY, 2006. ACM.

[84] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[85] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine learning*, pages 625–632. ACM, 2005.

[86] M. Ceci and D. Malerba. Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, 28(1):1–41, 2007.

[87] E. Costa, A. Lorena, A. Carvalho, A. A. Freitas, and N. Holden. Comparing several approaches for hierarchical classification of proteins with decision trees. In *Advances in Bioinformatics and Computational Biology*, volume 4643, pages 126–137, 2007.

[88] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research*, 7:31–54, 2006.

[89] H.-H. Huang. An ensemble distance measure of k-mer and natural vector for the phylogenetic analysis of multiple-segmented viruses. *Journal of Theoretical Biology*, 398:136–144, 2016.

[90] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4):427–437, 2009.

[91] E. P. Costa, A. C. Lorena, A. C. P. L. F. Carvalho, and A. A. Freitas. A review of performance evaluation measures for hierarchical classifiers. In C. Drummond, W. Elazmeh, N. Japkowicz, and S. A. Macskassy, editors, *Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop, AAAI Technical Report WS-07-05*, pages 182–196. AAAI Press, July 2007.

[92] A. Kosmopoulos, I. Partalas, E. Gaussier, G. Paliouras, and I. Androutsopoulos. Evaluation measures for hierarchical classification: a unified view and novel approaches. *Data Mining and Knowledge Discovery*, 29(3):820–865, 2015.

[93] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based

learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7(Jul):1601–1626, 2006.

[94] X.-Q. Li and D. Du. Variation, evolution, and correlation analysis of c+g content and genome or chromosome size in different kingdoms and phyla. *PLoS ONE*, 9(2), 2014.

[95] N. Wang, I. S. Mian, and M. Herbster. Predicting the Baltimore Group and ICTV Order of a virus using global statistics of whole genome sequence. *PLoS ONE*.

[96] J. L. Oliver and A. Marin. A relationship between gc content and coding-sequence length. *Journal of Molecular Evolution*, 43(3):216–223, 1996.

[97] H. Wu, Z. Zhang, S. Hu, and J. Yu. On the molecular mechanism of gc content variation among eubacterial genomes. *Biology Direct*, 7(2), 2012.

[98] J. A. Birdsell. Integrating genomics, bioinformatics, and classical genetics to study the effects of recombination on genome evolution. *Molecular Biology and Evolution*, 19(7):1181–1197, 2002.

[99] H.-H. Huang, C. Yu, H. Zheng, T. Hernandez, S.-C. Yau, R. L. He, J. Yang, and S. S.-T. Yau. Global comparison of multiple-segmented viruses in 12-dimensional genome space. *Molecular Phylogenetics and Evolution*, 81:29–36, 2014.

[100] G. Reinert, S. Schbath, and M. S. Waterman. Probabilistic and statistical properties of words: an overview. *Journal of Computational Biology*, 7(1-2):1–46, 2000.

[101] B. Ma, J. Tromp, and M. Li. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, 2002.

[102] S. R. Jun, G. E. Sims, G. A. Wu, and S. H. Kim. Whole-proteome phylogeny of prokaryotes by feature frequency profiles: an alignment-free method with

optimal feature resolution. *Proceedings of the National Academy of Sciences*, 107(1):133–138, 2010.

[103] I. F. Korf and A. B. Rose. Applying word-based algorithms: the IMEter. *Methods in Molecular Biology*, 553:287–301, 2009.

[104] G. E. Sims, S. R. Jun, G. A. Wu, and S. H. Kim. Alignment-free genome comparison with feature frequency profiles (ffp) and optimal resolutions. *Proceedings of the National Academy of Sciences*, 106:2677–2682, 2009.

[105] T. J. Wu, J. P. Burke, and D. B. Davison. A measure of DNA sequence dissimilarity based on Mahalanobis distance between frequencies of words. *Biometrics*, 53(4):1431–1439, 1997.

[106] B. Liu, L. Fang, R. Long, X. Lan, and K. C. Chou. ienhancer-2l: a two-layer predictor for identifying enhancers and their strength by pseudo k-tuple nucleotide composition. *Bioinformaitcs*, 32(3):362–369, 2016.

[107] B. Liu, L. Fang, J. Chen, F. Liu, and X. Wang. mirna-dis: microrna precursor identification based on distance structure status pairs. *Molecular BioSystems*, 11(4):1194–1204, 2015.

[108] K. E. Samocha, E. B. Robinson, S. J. Sanders, C. Stevens, A. Sabo, L. M. McGrath, J. A. Kosmicki, K. Rehnström, S. Mallick, A. Kirby, D. P. Wall, D. G. MacArthur, S. B. Gabriel, M. DePristo, S. M. Purcell, A. Palotie, E. Boerwinkle, J. D. Buxbaum, E. H. Cook, R. A. Gibbs, G. D. Schellenberg, J. S. Sutcliffe, B. Devlin, K. Roeder, B. M. Neale, and M. J. Daly. A framework for the interpretation of de novo mutation in human disease. *Nature Genetics*, 46(9):944–950, 2014.

[109] V. Aggarwala and B. F. Voight. An expanded sequence context model broadly explains variability in polymorphism levels across the human genome. *Nature Genetics*, 48(4):349–355, 2016.

[110] P. Compeau, P. Pevzner, and G. Teslar. How to apply de bruijn graphs to genome assembly? *Nature Biotechnology*, 29:987–991, 2011.

[111] D.R. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome Research*, 18(5):821–829, 2008.

[112] P. S. Kolekar, M. M. Kale, and U. Kulkarni-Kale. 'inter-arrival time' inspired algorithm and its application in clustering and molecular phylogeny. *AIP Conference Proceedings*, 1298:307–312, 2010.

[113] P. Kolekar, M. Kale, and U. Kulkarni-Kale. Alignment-free distance measure based on return time distribution for sequence analysis: Applications to clustering, molecular phylogeny and subtyping. *Molecular Phylogenetics and Evolution*, 65(2):510–522, 2012.

[114] R. Kumar, B. K. Mishra, T. Lahiri, G. Kumar, N. Kumar, R. Gupta, and M. K. Pal. Pcv: An alignment free method for finding homologous nucleotide sequences and its application in phylogenetic study. *Interdisciplinary Sciences*, pages 1–11, 2016.

[115] P. S. Kolekar, M. M. Kale, and U. Kulkarni-Kale. Genotyping of mumps viruses based on sh gene: development of a server using alignment-free and alignment-based methods. *Immunome Research*, 7(3):1–7, 2011.

[116] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: a string kernel for svm protein classification. In *Pacific Symposium on Biocomputing*, pages 564–575. World Scientific Publishing, 2002.

[117] G. Rätsch and S. Sonnenburg. *Kernel Methods in Computational Biology*, chapter Accurate Splice Site Detection for Caenorhabditis elegans, pages 277–298. The MIT press, 2004.

[118] E. Aurell, N. Innocenti, and H.-J. Zhou. The bulk and the tail of minimal absent words in genome sequences. *Physical Biology*, 13(2), 2016.

[119] S. P. Garcia, A. J. Pinho, J. M. O. S. Rodrigues, C. A. C. Bastos, and P. J. S. G. Ferreira. Minimal absent words in prokaryotic and eukaryotic genomes. *PLoS One*, 6(1), 2011.

[120] S. P. Garcia and A. J. Pinho. Minimal absent words in four human genome assemblies. *PLoS One*, 6(12), 2011.

[121] M. S. Rahman, A. Alatabbi, T. Athar, M. Crochemore, and M. S. Rahman. Absent words and the (dis)similarity analysis of dna sequences: an experimental study. *BMC Research Notes*, 9(186), 2016.

[122] Andrei N Kolmogorov. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 369–376, 1963.

[123] Vladimir V V'yugin. Algorithmic complexity and stochastic properties of finite binary sequences. *The Computer Journal*, 42(4):294–317, 1999.

[124] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[125] Michael Burrows and David J Wheeler. A block-sorting lossless data compression algorithm. 1994.

[126] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[127] A quick benchmark: Gzip vs. bzip2 vs. lzma, May 2015. Retrieved May 19, 2016, from https://tukaani.org/lzma/benchmarks.html.

[128] P. Deutsch. Deflate compressed data format specification, May 1996. Retrieved from November 1, 2016, from https://tools.ietf.org/html/rfc1951.

[129] N. S. Bakr and A. A. Sharawi. DNA lossless compression algorithms: Review. *American Journal of Bioinformatics Research*, 3(3):72–81, 2013.

[130] D. Pratas. *Compression and analysis of genomic data*. PhD thesis, Universidade de Aveiro, 2016.

[131] M. Sardaraz, M. Tahir, and A. A. Ikram. Advances in high throughput dna sequence data compression. *Journal of Bioinformatics and Computational Biology*, 14(3), 2016.

[132] M. Hosseini, D. Pratas, and A. J. Pinho. A survey on data compression methods for biological sequences. *Information*, 7(56), 2016.

[133] M. Mohammed, A. Dutta, T. Bose, S. Chadaram, and S. Mande. Deliminate - a fast and efficient method for loss-less compression of genomic sequences: Sequence analysis. *Bioinformatics*, 28(19):2527–2529, 2012.

[134] A. Pinho and D. Pratas. Mfcompress: A compression tool for fasta and multi-fasta data. *Bioinformatics*, 30(1):117–118, 2013.

[135] Armando J Pinho, Paulo JSG Ferreira, António JR Neves, and Carlos AC Bastos. On the representability of complete genomes by multiple competing finite-context (markov) models. *PLoS ONES*, 6(6):e21588, 2011.

[136] G. Benoit, C. Lemaitre, D. Lavenier, E. Drezen, T. Dayris, R. Uricaru, and G. Rizk. Reference-free compression of high throughput sequencing data with a probabilistic de bruijn graph. *BMC Bioinformatics*, 16(288), 2015.

[137] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[138] L. J. P. van der Maaten and G. E Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, Nov 2008.

[139] The r graphics package, May 2015. Retrieved November 19, 2015, from https://stat.ethz.ch/R-manual/R-devel/library/graphics/html/00Index.html.

[140] ggtern: An extension to 'ggplot2', for the creation of ternary diagrams, May 2015. Retrieved November 19, 2015, from https://cran.r-project.org/web/packages/ggtern/index.html.

[141] Rtsne: T-distributed stochastic neighbor embedding using barnes-hut implementation, May 2015. Retrieved November 19, 2015, from https://cran.r-project.org/web/packages/Rtsne/index.html.

[142] Prasert Auewarakul. Composition bias and genome polarity of rna viruses. *Virus Research*, 109(1):33–37, 2005.

[143] Ilya S Belalov and Alexander N Lukashev. Causes and implications of codon usage bias in rna viruses. *PLoS One*, 8(2):e56642, 2013.

[144] Formijn van Hemert, Antoinette C van der Kuyl, and Ben Berkhout. Impact of the biased nucleotide composition of viral rna genomes on rna structure and codon usage. *Journal of General Virology*, 97(10):2608–2619, 2016.

[145] C. Barton, A. Heliou, L. Mouchard, and S. P. Pissis. Linear-time computation of minimal absent words using suffix array. *BMC Bioinformatics*, 15(388), 2014.

[146] S. Chairungsee and M. Crochemore. Using minimal absent words to build phylogeny. *Theoretical Computer Science*, 450:109–116, 2012.

[147] A Kapoor, P Simmonds, WI Lipkin, S Zaidi, and E Delwart. Use of nucleotide composition analysis to infer hosts for three novel picorna-like viruses. *Journal of virology*, 84(19):10322–10328, 2010.

[148] Andrew B Goldberg and Xiaojin Zhu. Keepin'it real: semi-supervised learning with realistic tuning. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, pages 19–27. Association for Computational Linguistics, 2009.

[149] M. Bekkar and T. A. Alitouche. Imbalanced data learning approaches review. *International Journal of Data Mining and Knowledge Management Process*, 3(4):15–33, 2013.

[150] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[151] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

[152] C. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for svm protein classification. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002.

[153] S. Sonnenburg, G. Rätsch, and B. Schölkopf. Large scale genomic sequence svm classifiers. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 848–855, 2005.

[154] Daniel Kudenko and Haym Hirsh. Feature generation for sequence categorization. In *AAAI/IAAI*, pages 733–738, 1998.

[155] Neal Lesh, Mohammed J Zaki, and Mitsunori Ogihara. Mining features for sequence classification. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 342–346. ACM, 1999.

[156] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. Archive of Cornell University Library, September 2013.

[157] E. Asgari and M. R. K. Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLoS ONE*, 10(11), 2015.

[158] Qian Zhang, Se-Ran Jun, Michael Leuze, David Ussery, and Intawat Nookaew. Viral phylogenomics using an alignment-free method: A three-step approach to determine optimal length of k-mer. *Scientific reports*, 7:40712, 2017.

[159] Guozhu Dong and Jian Pei. *Sequence data mining*, volume 33. Springer Science and Business Media, 2007.

[160] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[161] O. Bonham-Carter, J. Steele, and D. Bastola. Alignment-free genetic sequence comparisons: a review of recent approaches by word analysis. *Briefings in Bioinformatics*, 15(6):890–905, 2014.

[162] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

[163] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484, 2005.

[164] Mark Herbster, Massimiliano Pontil, and Sergio R Galeano. Fast prediction on a tree. In *Advances in Neural Information Processing Systems*, pages 657–664, 2009.

[165] Giorgio Valentini. Hierarchical ensemble methods for protein function prediction. *ISRN bioinformatics*, 2014, 2014.

[166] J. Metz and A. A. Freitas. Extending hierarchical classification with semisupervised learning. In *Proceedings of the UK Workshop on Computational Intelligence*, pages 1–6, 2009.

[167] A. Santos and A. Canuto. Applying semi-supervised learning in hierarchical multi-label classification. *Expert Systems with Applications*, 41(14):6075–6085, 2014.

[168] Kristin P Bennett and Ayhan Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information processing systems*, pages 368–374, 1999.

[169] B. Dalvi, A. Mishra, and W. W. Cohen. Hierarchical semi-supervised classification with incomplete class hierarchies. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*, pages 193–202, 2016.